

# NVIDIA Performance Primitives (NPP)

Version 3.2.16

November 15, 2010



# Contents

<b>1</b>	<b>NVIDIA Performance Primitives</b>	<b>1</b>
1.1	What is NPP? . . . . .	1
1.2	Documentation . . . . .	1
1.3	Technical Specifications . . . . .	1
<b>2</b>	<b>General API Conventions</b>	<b>3</b>
2.1	Memory Management . . . . .	4
2.2	Function Naming . . . . .	4
<b>3</b>	<b>Signal-Processing Specific API Conventions</b>	<b>5</b>
3.1	Signal Data . . . . .	6
3.1.1	Parameter Names for Signal Data . . . . .	6
3.1.1.1	Source Signal Pointer . . . . .	6
3.1.1.2	Destination Signal Pointer . . . . .	6
3.1.1.3	In-Place Signal Pointer . . . . .	6
3.1.2	Signal Data Alignment Requirements . . . . .	7
3.1.3	Signal Data Related Error Codes . . . . .	7
3.2	Signal Length . . . . .	7
3.2.1	Length Related Error Codes . . . . .	7
<b>4</b>	<b>Imaging-Processing Specific API Conventions</b>	<b>9</b>
4.1	Function Naming . . . . .	10
4.2	Image Data . . . . .	10
4.2.1	Line Step . . . . .	11
4.2.2	Parameter Names for Image Data . . . . .	11
4.2.2.1	Passing Source-Image Data . . . . .	11
4.2.2.2	Passing Destination-Image Data . . . . .	12
4.2.2.3	Passing In-Place Image Data . . . . .	12
4.2.3	Image Data Alignment Requirements . . . . .	12

4.2.4	Image Data Related Error Codes . . . . .	13
4.3	Region-of-Interest (ROI) . . . . .	13
4.3.1	ROI Related Error Codes . . . . .	13
<b>5</b>	<b>Module Index</b>	<b>15</b>
5.1	Modules . . . . .	15
<b>6</b>	<b>Data Structure Index</b>	<b>17</b>
6.1	Data Structures . . . . .	17
<b>7</b>	<b>Module Documentation</b>	<b>19</b>
7.1	NPP Core . . . . .	19
7.1.1	Detailed Description . . . . .	19
7.1.2	Function Documentation . . . . .	19
7.1.2.1	nppGetGpuComputeCapability . . . . .	19
7.1.2.2	nppGetGpuName . . . . .	20
7.1.2.3	nppGetGpuNumSMs . . . . .	20
7.1.2.4	nppGetLibVersion . . . . .	20
7.1.2.5	nppGetMaxThreadsPerBlock . . . . .	20
7.2	NPP Type Definitions and Constants . . . . .	21
7.2.1	Define Documentation . . . . .	24
7.2.1.1	NPP_MAX_16S . . . . .	24
7.2.1.2	NPP_MAX_16U . . . . .	24
7.2.1.3	NPP_MAX_32S . . . . .	24
7.2.1.4	NPP_MAX_32U . . . . .	24
7.2.1.5	NPP_MAX_64S . . . . .	24
7.2.1.6	NPP_MAX_8S . . . . .	24
7.2.1.7	NPP_MAX_8U . . . . .	25
7.2.1.8	NPP_MAXABS_32F . . . . .	25
7.2.1.9	NPP_MAXABS_64F . . . . .	25
7.2.1.10	NPP_MIN_16S . . . . .	25
7.2.1.11	NPP_MIN_16U . . . . .	25
7.2.1.12	NPP_MIN_32S . . . . .	25
7.2.1.13	NPP_MIN_32U . . . . .	25
7.2.1.14	NPP_MIN_64S . . . . .	25
7.2.1.15	NPP_MIN_8S . . . . .	25
7.2.1.16	NPP_MIN_8U . . . . .	25
7.2.1.17	NPP_MINABS_32F . . . . .	25

7.2.1.18	NPP_MINABS_64F . . . . .	26
7.2.2	Enumeration Type Documentation . . . . .	26
7.2.2.1	NppCmpOp . . . . .	26
7.2.2.2	NppGpuComputeCapability . . . . .	26
7.2.2.3	NppHintAlgorithm . . . . .	26
7.2.2.4	NppiAxis . . . . .	26
7.2.2.5	NppiInterpolationMode . . . . .	27
7.2.2.6	NppRoundMode . . . . .	27
7.2.2.7	NppStatus . . . . .	27
7.3	Basic NPP Data Types . . . . .	29
7.3.1	Typedef Documentation . . . . .	30
7.3.1.1	Npp16s . . . . .	30
7.3.1.2	Npp16u . . . . .	30
7.3.1.3	Npp32f . . . . .	30
7.3.1.4	Npp32s . . . . .	30
7.3.1.5	Npp32u . . . . .	30
7.3.1.6	Npp64f . . . . .	30
7.3.1.7	Npp64s . . . . .	30
7.3.1.8	Npp64u . . . . .	30
7.3.1.9	Npp8s . . . . .	30
7.3.1.10	Npp8u . . . . .	30
7.4	NPP Image Processing . . . . .	31
7.4.1	Function Documentation . . . . .	70
7.4.1.1	nppiAbsDiff_32f_C1R . . . . .	70
7.4.1.2	nppiAbsDiff_32s_C1R . . . . .	70
7.4.1.3	nppiAbsDiff_8u_AC4R . . . . .	71
7.4.1.4	nppiAbsDiff_8u_C1R . . . . .	71
7.4.1.5	nppiAbsDiff_8u_C4R . . . . .	72
7.4.1.6	nppiAbsDiffC_32f_C1R . . . . .	72
7.4.1.7	nppiAdd_32f_C1R . . . . .	72
7.4.1.8	nppiAdd_32s_C1R . . . . .	73
7.4.1.9	nppiAdd_8u_AC4RSfs . . . . .	73
7.4.1.10	nppiAdd_8u_C1RSfs . . . . .	74
7.4.1.11	nppiAdd_8u_C4RSfs . . . . .	74
7.4.1.12	nppiAddC_32f_C1R . . . . .	74
7.4.1.13	nppiAddC_32fc_C1R . . . . .	75

7.4.1.14	<a href="#">nppiCanny_32f8u_C1R</a>	75
7.4.1.15	<a href="#">nppiCannyGetBufferSize</a>	76
7.4.1.16	<a href="#">nppiColorTwist32f_8u_AC4R</a>	76
7.4.1.17	<a href="#">nppiColorTwist32f_8u_C3R</a>	77
7.4.1.18	<a href="#">nppiColorTwist32f_8u_P3R</a>	77
7.4.1.19	<a href="#">nppiCompare_32f_C1R</a>	77
7.4.1.20	<a href="#">nppiCompare_8u_AC4R</a>	78
7.4.1.21	<a href="#">nppiCompare_8u_C4R</a>	78
7.4.1.22	<a href="#">nppiConvert_16s32f_C1R</a>	79
7.4.1.23	<a href="#">nppiConvert_16s32s_C1R</a>	79
7.4.1.24	<a href="#">nppiConvert_16s8u_AC4R</a>	80
7.4.1.25	<a href="#">nppiConvert_16s8u_C1R</a>	80
7.4.1.26	<a href="#">nppiConvert_16s8u_C4R</a>	80
7.4.1.27	<a href="#">nppiConvert_16u32f_C1R</a>	81
7.4.1.28	<a href="#">nppiConvert_16u32s_C1R</a>	81
7.4.1.29	<a href="#">nppiConvert_16u8u_AC4R</a>	81
7.4.1.30	<a href="#">nppiConvert_16u8u_C1R</a>	82
7.4.1.31	<a href="#">nppiConvert_16u8u_C4R</a>	82
7.4.1.32	<a href="#">nppiConvert_32f16s_C1R</a>	83
7.4.1.33	<a href="#">nppiConvert_32f16u_C1R</a>	83
7.4.1.34	<a href="#">nppiConvert_32f8u_C1R</a>	83
7.4.1.35	<a href="#">nppiConvert_8u16s_AC4R</a>	84
7.4.1.36	<a href="#">nppiConvert_8u16s_C1R</a>	84
7.4.1.37	<a href="#">nppiConvert_8u16s_C4R</a>	85
7.4.1.38	<a href="#">nppiConvert_8u16u_AC4R</a>	85
7.4.1.39	<a href="#">nppiConvert_8u16u_C1R</a>	85
7.4.1.40	<a href="#">nppiConvert_8u16u_C4R</a>	86
7.4.1.41	<a href="#">nppiConvert_8u32f_C1R</a>	86
7.4.1.42	<a href="#">nppiCopy_16s_AC4R</a>	86
7.4.1.43	<a href="#">nppiCopy_16s_C1R</a>	87
7.4.1.44	<a href="#">nppiCopy_16s_C4R</a>	87
7.4.1.45	<a href="#">nppiCopy_16u_AC4R</a>	87
7.4.1.46	<a href="#">nppiCopy_16u_C1R</a>	88
7.4.1.47	<a href="#">nppiCopy_16u_C4R</a>	88
7.4.1.48	<a href="#">nppiCopy_32f_AC4R</a>	88
7.4.1.49	<a href="#">nppiCopy_32f_C1R</a>	89

7.4.1.50	<a href="#">nppiCopy_32f_C4R</a>	89
7.4.1.51	<a href="#">nppiCopy_32s_AC4R</a>	89
7.4.1.52	<a href="#">nppiCopy_32s_C1R</a>	90
7.4.1.53	<a href="#">nppiCopy_32s_C4R</a>	90
7.4.1.54	<a href="#">nppiCopy_8u_AC4R</a>	90
7.4.1.55	<a href="#">nppiCopy_8u_C1R</a>	91
7.4.1.56	<a href="#">nppiCopy_8u_C4R</a>	91
7.4.1.57	<a href="#">nppiCopyConstBorder_32s_C1R</a>	91
7.4.1.58	<a href="#">nppiCopyConstBorder_8u_AC4R</a>	92
7.4.1.59	<a href="#">nppiCopyConstBorder_8u_C1R</a>	92
7.4.1.60	<a href="#">nppiCopyConstBorder_8u_C4R</a>	93
7.4.1.61	<a href="#">nppiDCTQuantFwd8x8LS_JPEG_8u16s_C1R</a>	93
7.4.1.62	<a href="#">nppiDCTQuantInv8x8LS_JPEG_16s8u_C1R</a>	94
7.4.1.63	<a href="#">nppiDilate_8u_C1R</a>	94
7.4.1.64	<a href="#">nppiDilate_8u_C4R</a>	95
7.4.1.65	<a href="#">nppiDiv_32f_C1R</a>	95
7.4.1.66	<a href="#">nppiDiv_32s_C1R</a>	96
7.4.1.67	<a href="#">nppiDiv_8u_AC4RSfs</a>	96
7.4.1.68	<a href="#">nppiDiv_8u_C1RSfs</a>	96
7.4.1.69	<a href="#">nppiDiv_8u_C4RSfs</a>	97
7.4.1.70	<a href="#">nppiDivC_32f_C1R</a>	97
7.4.1.71	<a href="#">nppiDivC_32fc_C1R</a>	98
7.4.1.72	<a href="#">nppiErode_8u_C1R</a>	98
7.4.1.73	<a href="#">nppiErode_8u_C4R</a>	99
7.4.1.74	<a href="#">nppiEvenLevelsHost_32s</a>	99
7.4.1.75	<a href="#">nppiExp_32f_C1R</a>	99
7.4.1.76	<a href="#">nppiFilter_8u_C1R</a>	100
7.4.1.77	<a href="#">nppiFilter_8u_C4R</a>	100
7.4.1.78	<a href="#">nppiFilterBox_8u_C1R</a>	101
7.4.1.79	<a href="#">nppiFilterBox_8u_C4R</a>	101
7.4.1.80	<a href="#">nppiFilterColumn_8u_C1R</a>	102
7.4.1.81	<a href="#">nppiFilterColumn_8u_C4R</a>	102
7.4.1.82	<a href="#">nppiFilterMax_8u_C1R</a>	103
7.4.1.83	<a href="#">nppiFilterMax_8u_C4R</a>	103
7.4.1.84	<a href="#">nppiFilterMin_8u_C1R</a>	104
7.4.1.85	<a href="#">nppiFilterMin_8u_C4R</a>	104

7.4.1.86	<a href="#">nppiFilterRow_8u_C1R</a>	105
7.4.1.87	<a href="#">nppiFilterRow_8u_C4R</a>	105
7.4.1.88	<a href="#">nppiFree</a>	106
7.4.1.89	<a href="#">nppiGetAffineBound</a>	106
7.4.1.90	<a href="#">nppiGetAffineQuad</a>	106
7.4.1.91	<a href="#">nppiGetAffineTransform</a>	107
7.4.1.92	<a href="#">nppiGetPerspectiveBound</a>	107
7.4.1.93	<a href="#">nppiGetPerspectiveQuad</a>	108
7.4.1.94	<a href="#">nppiGetPerspectiveTransform</a>	108
7.4.1.95	<a href="#">nppiGraphcut_32s8u</a>	108
7.4.1.96	<a href="#">nppiGraphcutGetSize</a>	109
7.4.1.97	<a href="#">nppiHistogramEven_16s_AC4R</a>	110
7.4.1.98	<a href="#">nppiHistogramEven_16s_C1R</a>	110
7.4.1.99	<a href="#">nppiHistogramEven_16s_C4R</a>	111
7.4.1.100	<a href="#">nppiHistogramEven_16u_AC4R</a>	111
7.4.1.101	<a href="#">nppiHistogramEven_16u_C1R</a>	112
7.4.1.102	<a href="#">nppiHistogramEven_16u_C4R</a>	112
7.4.1.103	<a href="#">nppiHistogramEven_8u_AC4R</a>	112
7.4.1.104	<a href="#">nppiHistogramEven_8u_C1R</a>	113
7.4.1.105	<a href="#">nppiHistogramEven_8u_C4R</a>	113
7.4.1.106	<a href="#">nppiHistogramEvenGetBufferSize_16s_AC4R</a>	114
7.4.1.107	<a href="#">nppiHistogramEvenGetBufferSize_16s_C1R</a>	114
7.4.1.108	<a href="#">nppiHistogramEvenGetBufferSize_16s_C4R</a>	114
7.4.1.109	<a href="#">nppiHistogramEvenGetBufferSize_16u_AC4R</a>	115
7.4.1.110	<a href="#">nppiHistogramEvenGetBufferSize_16u_C1R</a>	115
7.4.1.111	<a href="#">nppiHistogramEvenGetBufferSize_16u_C4R</a>	115
7.4.1.112	<a href="#">nppiHistogramEvenGetBufferSize_8u_AC4R</a>	116
7.4.1.113	<a href="#">nppiHistogramEvenGetBufferSize_8u_C1R</a>	116
7.4.1.114	<a href="#">nppiHistogramEvenGetBufferSize_8u_C4R</a>	116
7.4.1.115	<a href="#">nppiHistogramRange_16s_AC4R</a>	116
7.4.1.116	<a href="#">nppiHistogramRange_16s_C1R</a>	117
7.4.1.117	<a href="#">nppiHistogramRange_16s_C4R</a>	117
7.4.1.118	<a href="#">nppiHistogramRange_16u_AC4R</a>	118
7.4.1.119	<a href="#">nppiHistogramRange_16u_C1R</a>	118
7.4.1.120	<a href="#">nppiHistogramRange_16u_C4R</a>	119
7.4.1.121	<a href="#">nppiHistogramRange_32f_AC4R</a>	119



7.4.1.122	<a href="#">nppiHistogramRange_32f_C1R</a>	120
7.4.1.123	<a href="#">nppiHistogramRange_32f_C4R</a>	120
7.4.1.124	<a href="#">nppiHistogramRange_8u_AC4R</a>	120
7.4.1.125	<a href="#">nppiHistogramRange_8u_C1R</a>	121
7.4.1.126	<a href="#">nppiHistogramRange_8u_C4R</a>	121
7.4.1.127	<a href="#">nppiHistogramRangeGetBufferSize_16s_AC4R</a>	122
7.4.1.128	<a href="#">nppiHistogramRangeGetBufferSize_16s_C1R</a>	122
7.4.1.129	<a href="#">nppiHistogramRangeGetBufferSize_16s_C4R</a>	122
7.4.1.130	<a href="#">nppiHistogramRangeGetBufferSize_16u_AC4R</a>	123
7.4.1.131	<a href="#">nppiHistogramRangeGetBufferSize_16u_C1R</a>	123
7.4.1.132	<a href="#">nppiHistogramRangeGetBufferSize_16u_C4R</a>	123
7.4.1.133	<a href="#">nppiHistogramRangeGetBufferSize_32f_AC4R</a>	123
7.4.1.134	<a href="#">nppiHistogramRangeGetBufferSize_32f_C1R</a>	124
7.4.1.135	<a href="#">nppiHistogramRangeGetBufferSize_32f_C4R</a>	124
7.4.1.136	<a href="#">nppiHistogramRangeGetBufferSize_8u_AC4R</a>	124
7.4.1.137	<a href="#">nppiHistogramRangeGetBufferSize_8u_C1R</a>	125
7.4.1.138	<a href="#">nppiHistogramRangeGetBufferSize_8u_C4R</a>	125
7.4.1.139	<a href="#">nppiLn_32f_C1R</a>	125
7.4.1.140	<a href="#">nppiLUT_Linear_8u_AC4R</a>	126
7.4.1.141	<a href="#">nppiLUT_Linear_8u_C1R</a>	126
7.4.1.142	<a href="#">nppiLUT_Linear_8u_C3R</a>	127
7.4.1.143	<a href="#">nppiMagnitude_32fc32f_C1R</a>	127
7.4.1.144	<a href="#">nppiMagnitudeSqr_32fc32f_C1R</a>	127
7.4.1.145	<a href="#">nppiMalloc_16s_C1</a>	128
7.4.1.146	<a href="#">nppiMalloc_16s_C4</a>	128
7.4.1.147	<a href="#">nppiMalloc_16u_C1</a>	128
7.4.1.148	<a href="#">nppiMalloc_16u_C3</a>	129
7.4.1.149	<a href="#">nppiMalloc_16u_C4</a>	129
7.4.1.150	<a href="#">nppiMalloc_32f_C1</a>	129
7.4.1.151	<a href="#">nppiMalloc_32f_C2</a>	130
7.4.1.152	<a href="#">nppiMalloc_32f_C3</a>	130
7.4.1.153	<a href="#">nppiMalloc_32f_C4</a>	130
7.4.1.154	<a href="#">nppiMalloc_32s_C1</a>	130
7.4.1.155	<a href="#">nppiMalloc_32s_C3</a>	131
7.4.1.156	<a href="#">nppiMalloc_32s_C4</a>	131
7.4.1.157	<a href="#">nppiMalloc_8u_C1</a>	131

7.4.1.158	<a href="#">nppiMalloc_8u_C2</a>	132
7.4.1.159	<a href="#">nppiMalloc_8u_C3</a>	132
7.4.1.160	<a href="#">nppiMalloc_8u_C4</a>	132
7.4.1.161	<a href="#">nppiMean_StdDev_8u_C1R</a>	132
7.4.1.162	<a href="#">nppiMinMax_8u_C1R</a>	133
7.4.1.163	<a href="#">nppiMinMax_8u_C4R</a>	133
7.4.1.164	<a href="#">nppiMirror_8u_C1R</a>	134
7.4.1.165	<a href="#">nppiMirror_8u_C4R</a>	134
7.4.1.166	<a href="#">nppiMul_32f_C1R</a>	134
7.4.1.167	<a href="#">nppiMul_32s_C1R</a>	135
7.4.1.168	<a href="#">nppiMul_8u_AC4RSfs</a>	135
7.4.1.169	<a href="#">nppiMul_8u_C1RSfs</a>	136
7.4.1.170	<a href="#">nppiMul_8u_C4RSfs</a>	136
7.4.1.171	<a href="#">nppiMulC_32f_C1R</a>	136
7.4.1.172	<a href="#">nppiMulC_32fc_C1R</a>	137
7.4.1.173	<a href="#">nppiNormDiff_Inf_8u_C1R</a>	137
7.4.1.174	<a href="#">nppiNormDiff_L1_8u_C1R</a>	138
7.4.1.175	<a href="#">nppiNormDiff_L2_8u_C1R</a>	138
7.4.1.176	<a href="#">nppiQuantFwdRawTableInit_JPEG_8u</a>	138
7.4.1.177	<a href="#">nppiQuantFwdTableInit_JPEG_8u16u</a>	139
7.4.1.178	<a href="#">nppiQuantInvTableInit_JPEG_8u16u</a>	139
7.4.1.179	<a href="#">nppiRectStdDev_32s32f_C1R</a>	139
7.4.1.180	<a href="#">nppiReductionGetBufferHostSize_8u_C1R</a>	140
7.4.1.181	<a href="#">nppiReductionGetBufferHostSize_8u_C4R</a>	140
7.4.1.182	<a href="#">nppiResize_8u_C1R</a>	140
7.4.1.183	<a href="#">nppiResize_8u_C4R</a>	141
7.4.1.184	<a href="#">nppiRGBToYCbCr420_8u_C3P3R</a>	141
7.4.1.185	<a href="#">nppiRGBToYCbCr422_8u_C3C2R</a>	142
7.4.1.186	<a href="#">nppiRGBToYCbCr_8u_AC4R</a>	142
7.4.1.187	<a href="#">nppiRGBToYCbCr_8u_C3R</a>	143
7.4.1.188	<a href="#">nppiRGBToYCbCr_8u_P3R</a>	143
7.4.1.189	<a href="#">nppiRotate_8u_C1R</a>	143
7.4.1.190	<a href="#">nppiRotate_8u_C4R</a>	144
7.4.1.191	<a href="#">nppiSet_16s_AC4MR</a>	145
7.4.1.192	<a href="#">nppiSet_16s_AC4R</a>	145
7.4.1.193	<a href="#">nppiSet_16s_C1MR</a>	145

7.4.1.194 nppiSet_16s_C1R . . . . .	146
7.4.1.195 nppiSet_16s_C2R . . . . .	146
7.4.1.196 nppiSet_16s_C4CR . . . . .	146
7.4.1.197 nppiSet_16s_C4MR . . . . .	147
7.4.1.198 nppiSet_16s_C4R . . . . .	147
7.4.1.199 nppiSet_16u_AC4MR . . . . .	148
7.4.1.200 nppiSet_16u_AC4R . . . . .	148
7.4.1.201 nppiSet_16u_C1MR . . . . .	148
7.4.1.202 nppiSet_16u_C1R . . . . .	149
7.4.1.203 nppiSet_16u_C2R . . . . .	149
7.4.1.204 nppiSet_16u_C4CR . . . . .	149
7.4.1.205 nppiSet_16u_C4MR . . . . .	150
7.4.1.206 nppiSet_16u_C4R . . . . .	150
7.4.1.207 nppiSet_32f_AC4MR . . . . .	151
7.4.1.208 nppiSet_32f_AC4R . . . . .	151
7.4.1.209 nppiSet_32f_C1MR . . . . .	151
7.4.1.210 nppiSet_32f_C1R . . . . .	152
7.4.1.211 nppiSet_32f_C4CR . . . . .	152
7.4.1.212 nppiSet_32f_C4MR . . . . .	152
7.4.1.213 nppiSet_32f_C4R . . . . .	153
7.4.1.214 nppiSet_32s_AC4MR . . . . .	153
7.4.1.215 nppiSet_32s_AC4R . . . . .	154
7.4.1.216 nppiSet_32s_C1MR . . . . .	154
7.4.1.217 nppiSet_32s_C1R . . . . .	154
7.4.1.218 nppiSet_32s_C4CR . . . . .	155
7.4.1.219 nppiSet_32s_C4MR . . . . .	155
7.4.1.220 nppiSet_32s_C4R . . . . .	155
7.4.1.221 nppiSet_8u_AC4MR . . . . .	156
7.4.1.222 nppiSet_8u_AC4R . . . . .	156
7.4.1.223 nppiSet_8u_C1MR . . . . .	157
7.4.1.224 nppiSet_8u_C1R . . . . .	157
7.4.1.225 nppiSet_8u_C4CR . . . . .	157
7.4.1.226 nppiSet_8u_C4MR . . . . .	158
7.4.1.227 nppiSet_8u_C4R . . . . .	158
7.4.1.228 nppiSetDefaultQuantTable . . . . .	158
7.4.1.229 nppiSqrIntegral_8u32s32f_C1R . . . . .	159

7.4.1.230 nppiSub_32f_C1R . . . . .	159
7.4.1.231 nppiSub_32s_C1R . . . . .	160
7.4.1.232 nppiSub_8u_AC4RSfs . . . . .	160
7.4.1.233 nppiSub_8u_C1RSfs . . . . .	160
7.4.1.234 nppiSub_8u_C4RSfs . . . . .	161
7.4.1.235 nppiSubC_32f_C1R . . . . .	161
7.4.1.236 nppiSubC_32fc_C1R . . . . .	162
7.4.1.237 nppiSum_8u_C1R . . . . .	162
7.4.1.238 nppiSum_8u_C4R . . . . .	162
7.4.1.239 nppiSumWindowColumn_8u32f_C1R . . . . .	163
7.4.1.240 nppiSumWindowRow_8u32f_C1R . . . . .	163
7.4.1.241 nppiSwapChannels_8u_C4IR . . . . .	164
7.4.1.242 nppiThreshold_32f_C1R . . . . .	164
7.4.1.243 nppiThreshold_8u_AC4R . . . . .	165
7.4.1.244 nppiTranspose_8u_C1R . . . . .	165
7.4.1.245 nppiWarpAffine_16u_AC4R . . . . .	165
7.4.1.246 nppiWarpAffine_16u_C1R . . . . .	166
7.4.1.247 nppiWarpAffine_16u_C3R . . . . .	167
7.4.1.248 nppiWarpAffine_16u_C4R . . . . .	167
7.4.1.249 nppiWarpAffine_16u_P3R . . . . .	167
7.4.1.250 nppiWarpAffine_16u_P4R . . . . .	167
7.4.1.251 nppiWarpAffine_32f_AC4R . . . . .	168
7.4.1.252 nppiWarpAffine_32f_C1R . . . . .	168
7.4.1.253 nppiWarpAffine_32f_C3R . . . . .	169
7.4.1.254 nppiWarpAffine_32f_C4R . . . . .	169
7.4.1.255 nppiWarpAffine_32f_P3R . . . . .	169
7.4.1.256 nppiWarpAffine_32f_P4R . . . . .	169
7.4.1.257 nppiWarpAffine_32s_AC4R . . . . .	170
7.4.1.258 nppiWarpAffine_32s_C1R . . . . .	170
7.4.1.259 nppiWarpAffine_32s_C3R . . . . .	171
7.4.1.260 nppiWarpAffine_32s_C4R . . . . .	171
7.4.1.261 nppiWarpAffine_32s_P3R . . . . .	171
7.4.1.262 nppiWarpAffine_32s_P4R . . . . .	171
7.4.1.263 nppiWarpAffine_8u_AC4R . . . . .	172
7.4.1.264 nppiWarpAffine_8u_C1R . . . . .	172
7.4.1.265 nppiWarpAffine_8u_C3R . . . . .	173

7.4.1.266 nppiWarpAffine_8u_C4R . . . . .	173
7.4.1.267 nppiWarpAffine_8u_P3R . . . . .	173
7.4.1.268 nppiWarpAffine_8u_P4R . . . . .	174
7.4.1.269 nppiWarpAffineBack_16u_AC4R . . . . .	174
7.4.1.270 nppiWarpAffineBack_16u_C1R . . . . .	174
7.4.1.271 nppiWarpAffineBack_16u_C3R . . . . .	175
7.4.1.272 nppiWarpAffineBack_16u_C4R . . . . .	175
7.4.1.273 nppiWarpAffineBack_16u_P3R . . . . .	176
7.4.1.274 nppiWarpAffineBack_16u_P4R . . . . .	176
7.4.1.275 nppiWarpAffineBack_32f_AC4R . . . . .	176
7.4.1.276 nppiWarpAffineBack_32f_C1R . . . . .	176
7.4.1.277 nppiWarpAffineBack_32f_C3R . . . . .	177
7.4.1.278 nppiWarpAffineBack_32f_C4R . . . . .	177
7.4.1.279 nppiWarpAffineBack_32f_P3R . . . . .	178
7.4.1.280 nppiWarpAffineBack_32f_P4R . . . . .	178
7.4.1.281 nppiWarpAffineBack_32s_AC4R . . . . .	178
7.4.1.282 nppiWarpAffineBack_32s_C1R . . . . .	178
7.4.1.283 nppiWarpAffineBack_32s_C3R . . . . .	179
7.4.1.284 nppiWarpAffineBack_32s_C4R . . . . .	179
7.4.1.285 nppiWarpAffineBack_32s_P3R . . . . .	180
7.4.1.286 nppiWarpAffineBack_32s_P4R . . . . .	180
7.4.1.287 nppiWarpAffineBack_8u_AC4R . . . . .	180
7.4.1.288 nppiWarpAffineBack_8u_C1R . . . . .	180
7.4.1.289 nppiWarpAffineBack_8u_C3R . . . . .	181
7.4.1.290 nppiWarpAffineBack_8u_C4R . . . . .	182
7.4.1.291 nppiWarpAffineBack_8u_P3R . . . . .	182
7.4.1.292 nppiWarpAffineBack_8u_P4R . . . . .	182
7.4.1.293 nppiWarpAffineQuad_16u_AC4R . . . . .	182
7.4.1.294 nppiWarpAffineQuad_16u_C1R . . . . .	182
7.4.1.295 nppiWarpAffineQuad_16u_C3R . . . . .	183
7.4.1.296 nppiWarpAffineQuad_16u_C4R . . . . .	184
7.4.1.297 nppiWarpAffineQuad_16u_P3R . . . . .	184
7.4.1.298 nppiWarpAffineQuad_16u_P4R . . . . .	184
7.4.1.299 nppiWarpAffineQuad_32f_AC4R . . . . .	184
7.4.1.300 nppiWarpAffineQuad_32f_C1R . . . . .	184
7.4.1.301 nppiWarpAffineQuad_32f_C3R . . . . .	185

7.4.1.302 nppiWarpAffineQuad_32f_C4R . . . . .	185
7.4.1.303 nppiWarpAffineQuad_32f_P3R . . . . .	186
7.4.1.304 nppiWarpAffineQuad_32f_P4R . . . . .	186
7.4.1.305 nppiWarpAffineQuad_32s_AC4R . . . . .	186
7.4.1.306 nppiWarpAffineQuad_32s_C1R . . . . .	186
7.4.1.307 nppiWarpAffineQuad_32s_C3R . . . . .	187
7.4.1.308 nppiWarpAffineQuad_32s_C4R . . . . .	187
7.4.1.309 nppiWarpAffineQuad_32s_P3R . . . . .	187
7.4.1.310 nppiWarpAffineQuad_32s_P4R . . . . .	188
7.4.1.311 nppiWarpAffineQuad_8u_AC4R . . . . .	188
7.4.1.312 nppiWarpAffineQuad_8u_C1R . . . . .	188
7.4.1.313 nppiWarpAffineQuad_8u_C3R . . . . .	189
7.4.1.314 nppiWarpAffineQuad_8u_C4R . . . . .	189
7.4.1.315 nppiWarpAffineQuad_8u_P3R . . . . .	189
7.4.1.316 nppiWarpAffineQuad_8u_P4R . . . . .	190
7.4.1.317 nppiWarpPerspective_16u_AC4R . . . . .	190
7.4.1.318 nppiWarpPerspective_16u_C1R . . . . .	190
7.4.1.319 nppiWarpPerspective_16u_C3R . . . . .	191
7.4.1.320 nppiWarpPerspective_16u_C4R . . . . .	191
7.4.1.321 nppiWarpPerspective_16u_P3R . . . . .	192
7.4.1.322 nppiWarpPerspective_16u_P4R . . . . .	192
7.4.1.323 nppiWarpPerspective_32f_AC4R . . . . .	192
7.4.1.324 nppiWarpPerspective_32f_C1R . . . . .	192
7.4.1.325 nppiWarpPerspective_32f_C3R . . . . .	193
7.4.1.326 nppiWarpPerspective_32f_C4R . . . . .	193
7.4.1.327 nppiWarpPerspective_32f_P3R . . . . .	194
7.4.1.328 nppiWarpPerspective_32f_P4R . . . . .	194
7.4.1.329 nppiWarpPerspective_32s_AC4R . . . . .	194
7.4.1.330 nppiWarpPerspective_32s_C1R . . . . .	194
7.4.1.331 nppiWarpPerspective_32s_C3R . . . . .	195
7.4.1.332 nppiWarpPerspective_32s_C4R . . . . .	195
7.4.1.333 nppiWarpPerspective_32s_P3R . . . . .	196
7.4.1.334 nppiWarpPerspective_32s_P4R . . . . .	196
7.4.1.335 nppiWarpPerspective_8u_AC4R . . . . .	196
7.4.1.336 nppiWarpPerspective_8u_C1R . . . . .	196
7.4.1.337 nppiWarpPerspective_8u_C3R . . . . .	197

7.4.1.338 nppiWarpPerspective_8u_C4R . . . . .	198
7.4.1.339 nppiWarpPerspective_8u_P3R . . . . .	198
7.4.1.340 nppiWarpPerspective_8u_P4R . . . . .	198
7.4.1.341 nppiWarpPerspectiveBack_16u_AC4R . . . . .	198
7.4.1.342 nppiWarpPerspectiveBack_16u_C1R . . . . .	198
7.4.1.343 nppiWarpPerspectiveBack_16u_C3R . . . . .	200
7.4.1.344 nppiWarpPerspectiveBack_16u_C4R . . . . .	200
7.4.1.345 nppiWarpPerspectiveBack_16u_P3R . . . . .	200
7.4.1.346 nppiWarpPerspectiveBack_16u_P4R . . . . .	200
7.4.1.347 nppiWarpPerspectiveBack_32f_AC4R . . . . .	200
7.4.1.348 nppiWarpPerspectiveBack_32f_C1R . . . . .	201
7.4.1.349 nppiWarpPerspectiveBack_32f_C3R . . . . .	202
7.4.1.350 nppiWarpPerspectiveBack_32f_C4R . . . . .	202
7.4.1.351 nppiWarpPerspectiveBack_32f_P3R . . . . .	202
7.4.1.352 nppiWarpPerspectiveBack_32f_P4R . . . . .	202
7.4.1.353 nppiWarpPerspectiveBack_32s_AC4R . . . . .	202
7.4.1.354 nppiWarpPerspectiveBack_32s_C1R . . . . .	203
7.4.1.355 nppiWarpPerspectiveBack_32s_C3R . . . . .	204
7.4.1.356 nppiWarpPerspectiveBack_32s_C4R . . . . .	204
7.4.1.357 nppiWarpPerspectiveBack_32s_P3R . . . . .	204
7.4.1.358 nppiWarpPerspectiveBack_32s_P4R . . . . .	204
7.4.1.359 nppiWarpPerspectiveBack_8u_AC4R . . . . .	204
7.4.1.360 nppiWarpPerspectiveBack_8u_C1R . . . . .	205
7.4.1.361 nppiWarpPerspectiveBack_8u_C3R . . . . .	206
7.4.1.362 nppiWarpPerspectiveBack_8u_C4R . . . . .	206
7.4.1.363 nppiWarpPerspectiveBack_8u_P3R . . . . .	206
7.4.1.364 nppiWarpPerspectiveBack_8u_P4R . . . . .	206
7.4.1.365 nppiWarpPerspectiveQuad_16u_AC4R . . . . .	207
7.4.1.366 nppiWarpPerspectiveQuad_16u_C1R . . . . .	207
7.4.1.367 nppiWarpPerspectiveQuad_16u_C3R . . . . .	208
7.4.1.368 nppiWarpPerspectiveQuad_16u_C4R . . . . .	208
7.4.1.369 nppiWarpPerspectiveQuad_16u_P3R . . . . .	208
7.4.1.370 nppiWarpPerspectiveQuad_16u_P4R . . . . .	208
7.4.1.371 nppiWarpPerspectiveQuad_32f_AC4R . . . . .	209
7.4.1.372 nppiWarpPerspectiveQuad_32f_C1R . . . . .	209
7.4.1.373 nppiWarpPerspectiveQuad_32f_C3R . . . . .	210

7.4.1.374	<a href="#">nppiWarpPerspectiveQuad_32f_C4R</a>	210
7.4.1.375	<a href="#">nppiWarpPerspectiveQuad_32f_P3R</a>	210
7.4.1.376	<a href="#">nppiWarpPerspectiveQuad_32f_P4R</a>	210
7.4.1.377	<a href="#">nppiWarpPerspectiveQuad_32s_AC4R</a>	210
7.4.1.378	<a href="#">nppiWarpPerspectiveQuad_32s_C1R</a>	211
7.4.1.379	<a href="#">nppiWarpPerspectiveQuad_32s_C3R</a>	211
7.4.1.380	<a href="#">nppiWarpPerspectiveQuad_32s_C4R</a>	212
7.4.1.381	<a href="#">nppiWarpPerspectiveQuad_32s_P3R</a>	212
7.4.1.382	<a href="#">nppiWarpPerspectiveQuad_32s_P4R</a>	212
7.4.1.383	<a href="#">nppiWarpPerspectiveQuad_8u_AC4R</a>	212
7.4.1.384	<a href="#">nppiWarpPerspectiveQuad_8u_C1R</a>	212
7.4.1.385	<a href="#">nppiWarpPerspectiveQuad_8u_C3R</a>	213
7.4.1.386	<a href="#">nppiWarpPerspectiveQuad_8u_C4R</a>	214
7.4.1.387	<a href="#">nppiWarpPerspectiveQuad_8u_P3R</a>	214
7.4.1.388	<a href="#">nppiWarpPerspectiveQuad_8u_P4R</a>	214
7.4.1.389	<a href="#">nppiYCbCr420ToRGB_8u_P3C3R</a>	214
7.4.1.390	<a href="#">nppiYCbCr420ToYCbCr411_8u_P3P2R</a>	215
7.4.1.391	<a href="#">nppiYCbCr420ToYCbCr422_8u_P3R</a>	215
7.4.1.392	<a href="#">nppiYCbCr422ToRGB_8u_C2C3R</a>	215
7.4.1.393	<a href="#">nppiYCbCr422ToYCbCr411_8u_P3R</a>	216
7.4.1.394	<a href="#">nppiYCbCr422ToYCbCr420_8u_P3R</a>	216
7.4.1.395	<a href="#">nppiYCbCrToRGB_8u_AC4R</a>	216
7.4.1.396	<a href="#">nppiYCbCrToRGB_8u_C3R</a>	217
7.4.1.397	<a href="#">nppiYCbCrToRGB_8u_P3R</a>	217
7.5	<a href="#">NPP Signal Processing</a>	218
7.5.1	<a href="#">Function Documentation</a>	224
7.5.1.1	<a href="#">nppsCopy_16s</a>	224
7.5.1.2	<a href="#">nppsCopy_16sc</a>	225
7.5.1.3	<a href="#">nppsCopy_32f</a>	225
7.5.1.4	<a href="#">nppsCopy_32fc</a>	225
7.5.1.5	<a href="#">nppsCopy_32s</a>	226
7.5.1.6	<a href="#">nppsCopy_32sc</a>	226
7.5.1.7	<a href="#">nppsCopy_64fc</a>	226
7.5.1.8	<a href="#">nppsCopy_64s</a>	226
7.5.1.9	<a href="#">nppsCopy_64sc</a>	227
7.5.1.10	<a href="#">nppsCopy_8u</a>	227



7.5.1.11	<a href="#">nppsFree</a>	227
7.5.1.12	<a href="#">nppsMalloc_16s</a>	227
7.5.1.13	<a href="#">nppsMalloc_16sc</a>	228
7.5.1.14	<a href="#">nppsMalloc_16u</a>	228
7.5.1.15	<a href="#">nppsMalloc_32f</a>	228
7.5.1.16	<a href="#">nppsMalloc_32fc</a>	228
7.5.1.17	<a href="#">nppsMalloc_32s</a>	229
7.5.1.18	<a href="#">nppsMalloc_32sc</a>	229
7.5.1.19	<a href="#">nppsMalloc_32u</a>	229
7.5.1.20	<a href="#">nppsMalloc_64f</a>	229
7.5.1.21	<a href="#">nppsMalloc_64fc</a>	230
7.5.1.22	<a href="#">nppsMalloc_64s</a>	230
7.5.1.23	<a href="#">nppsMalloc_64sc</a>	230
7.5.1.24	<a href="#">nppsMalloc_8u</a>	230
7.5.1.25	<a href="#">nppsMax_16s</a>	231
7.5.1.26	<a href="#">nppsMax_32f</a>	231
7.5.1.27	<a href="#">nppsMax_32s</a>	231
7.5.1.28	<a href="#">nppsMax_64f</a>	232
7.5.1.29	<a href="#">nppsMin_16s</a>	232
7.5.1.30	<a href="#">nppsMin_32f</a>	232
7.5.1.31	<a href="#">nppsMin_32s</a>	233
7.5.1.32	<a href="#">nppsMin_64f</a>	233
7.5.1.33	<a href="#">nppsMinMax_16s</a>	233
7.5.1.34	<a href="#">nppsMinMax_16u</a>	234
7.5.1.35	<a href="#">nppsMinMax_32f</a>	234
7.5.1.36	<a href="#">nppsMinMax_32s</a>	234
7.5.1.37	<a href="#">nppsMinMax_32u</a>	235
7.5.1.38	<a href="#">nppsMinMax_64f</a>	235
7.5.1.39	<a href="#">nppsMinMax_8u</a>	235
7.5.1.40	<a href="#">nppsMinMaxGetBufferSize_16s</a>	236
7.5.1.41	<a href="#">nppsMinMaxGetBufferSize_16u</a>	236
7.5.1.42	<a href="#">nppsMinMaxGetBufferSize_32f</a>	236
7.5.1.43	<a href="#">nppsMinMaxGetBufferSize_32s</a>	236
7.5.1.44	<a href="#">nppsMinMaxGetBufferSize_32u</a>	237
7.5.1.45	<a href="#">nppsMinMaxGetBufferSize_64f</a>	237
7.5.1.46	<a href="#">nppsMinMaxGetBufferSize_8u</a>	237

7.5.1.47	<a href="#">nppsReductionGetBufferSize_16s</a>	237
7.5.1.48	<a href="#">nppsReductionGetBufferSize_16s_Sfs</a>	238
7.5.1.49	<a href="#">nppsReductionGetBufferSize_16sc</a>	238
7.5.1.50	<a href="#">nppsReductionGetBufferSize_16sc_Sfs</a>	238
7.5.1.51	<a href="#">nppsReductionGetBufferSize_16u</a>	238
7.5.1.52	<a href="#">nppsReductionGetBufferSize_32f</a>	239
7.5.1.53	<a href="#">nppsReductionGetBufferSize_32fc</a>	239
7.5.1.54	<a href="#">nppsReductionGetBufferSize_32s</a>	239
7.5.1.55	<a href="#">nppsReductionGetBufferSize_32s_Sfs</a>	240
7.5.1.56	<a href="#">nppsReductionGetBufferSize_32sc</a>	240
7.5.1.57	<a href="#">nppsReductionGetBufferSize_32u</a>	240
7.5.1.58	<a href="#">nppsReductionGetBufferSize_64f</a>	240
7.5.1.59	<a href="#">nppsReductionGetBufferSize_64fc</a>	241
7.5.1.60	<a href="#">nppsReductionGetBufferSize_64s</a>	241
7.5.1.61	<a href="#">nppsReductionGetBufferSize_8u</a>	241
7.5.1.62	<a href="#">nppsSet_16s</a>	242
7.5.1.63	<a href="#">nppsSet_16sc</a>	242
7.5.1.64	<a href="#">nppsSet_32f</a>	242
7.5.1.65	<a href="#">nppsSet_32fc</a>	242
7.5.1.66	<a href="#">nppsSet_32s</a>	243
7.5.1.67	<a href="#">nppsSet_32sc</a>	243
7.5.1.68	<a href="#">nppsSet_64f</a>	243
7.5.1.69	<a href="#">nppsSet_64fc</a>	244
7.5.1.70	<a href="#">nppsSet_64s</a>	244
7.5.1.71	<a href="#">nppsSet_64sc</a>	244
7.5.1.72	<a href="#">nppsSet_8u</a>	244
7.5.1.73	<a href="#">nppsSum_16s32s_Sfs</a>	245
7.5.1.74	<a href="#">nppsSum_16s_Sfs</a>	245
7.5.1.75	<a href="#">nppsSum_16sc32sc_Sfs</a>	245
7.5.1.76	<a href="#">nppsSum_16sc_Sfs</a>	246
7.5.1.77	<a href="#">nppsSum_32f</a>	246
7.5.1.78	<a href="#">nppsSum_32fc</a>	246
7.5.1.79	<a href="#">nppsSum_32s_Sfs</a>	247
7.5.1.80	<a href="#">nppsSum_64f</a>	247
7.5.1.81	<a href="#">nppsSum_64fc</a>	247
7.5.1.82	<a href="#">nppsZero_16s</a>	248

7.5.1.83	nppsZero_16sc . . . . .	248
7.5.1.84	nppsZero_32f . . . . .	248
7.5.1.85	nppsZero_32fc . . . . .	249
7.5.1.86	nppsZero_32s . . . . .	249
7.5.1.87	nppsZero_32sc . . . . .	249
7.5.1.88	nppsZero_64f . . . . .	249
7.5.1.89	nppsZero_64fc . . . . .	250
7.5.1.90	nppsZero_64s . . . . .	250
7.5.1.91	nppsZero_64sc . . . . .	250
7.5.1.92	nppsZero_8u . . . . .	250
<b>8</b>	<b>Data Structure Documentation</b>	<b>251</b>
8.1	Npp16sc Struct Reference . . . . .	251
8.1.1	Detailed Description . . . . .	251
8.1.2	Field Documentation . . . . .	251
8.1.2.1	im . . . . .	251
8.1.2.2	re . . . . .	251
8.2	Npp32fc Struct Reference . . . . .	252
8.2.1	Detailed Description . . . . .	252
8.2.2	Field Documentation . . . . .	252
8.2.2.1	im . . . . .	252
8.2.2.2	re . . . . .	252
8.3	Npp32sc Struct Reference . . . . .	253
8.3.1	Detailed Description . . . . .	253
8.3.2	Field Documentation . . . . .	253
8.3.2.1	im . . . . .	253
8.3.2.2	re . . . . .	253
8.4	Npp64fc Struct Reference . . . . .	254
8.4.1	Detailed Description . . . . .	254
8.4.2	Field Documentation . . . . .	254
8.4.2.1	im . . . . .	254
8.4.2.2	re . . . . .	254
8.5	Npp64sc Struct Reference . . . . .	255
8.5.1	Detailed Description . . . . .	255
8.5.2	Field Documentation . . . . .	255
8.5.2.1	im . . . . .	255
8.5.2.2	re . . . . .	255

8.6	NppiHaarBuffer Struct Reference	256
8.6.1	Field Documentation	256
8.6.1.1	haarBuffer	256
8.6.1.2	haarBufferSize	256
8.7	NppiHaarClassifier_32f Struct Reference	257
8.7.1	Field Documentation	257
8.7.1.1	classifiers	257
8.7.1.2	classifierSize	257
8.7.1.3	classifierStep	257
8.7.1.4	counterDevice	257
8.7.1.5	numClassifiers	257
8.8	NppiPoint Struct Reference	258
8.8.1	Detailed Description	258
8.8.2	Field Documentation	258
8.8.2.1	x	258
8.8.2.2	y	258
8.9	NppiRect Struct Reference	259
8.9.1	Detailed Description	259
8.9.2	Field Documentation	259
8.9.2.1	height	259
8.9.2.2	width	259
8.9.2.3	x	259
8.9.2.4	y	259
8.10	NppiSize Struct Reference	260
8.10.1	Detailed Description	260
8.10.2	Field Documentation	260
8.10.2.1	height	260
8.10.2.2	width	260
8.11	NppLibraryVersion Struct Reference	261
8.11.1	Field Documentation	261
8.11.1.1	build	261
8.11.1.2	major	261
8.11.1.3	minor	261

# Chapter 1

## NVIDIA Performance Primitives

### 1.1 What is NPP?

NVIDIA NPP is a library of functions for performing CUDA accelerated processing. The initial set of functionality in the library focuses on imaging and video processing and is widely applicable for developers in these areas. NPP will evolve over time to encompass more of the compute heavy tasks in a variety of problem domains. The NPP library is written to maximize flexibility, while maintaining high performance.

NPP can be used in one of two ways:

- A stand-alone library for adding GPU acceleration to an application with minimal effort. Using this route allows developers to add GPU acceleration to their applications in a matter of hours.
- A cooperative library for interoperating with a developer's GPU code efficiently.

Either route allows developers to harness the massive compute resources of NVIDIA GPUs, while simultaneously reducing development times.

### 1.2 Documentation

- [General API Conventions](#)
- [Signal-Processing Specific API Conventions](#)
- [Imaging-Processing Specific API Conventions](#)

### 1.3 Technical Specifications

Supported Platforms:

- Microsoft Windows 7 (64-bit and 32-bit)
- Microsoft Windows Vista (64-bit and 32-bit)
- Microsoft Windows XP (64-bit and 32-bit)
- Linux (Centos & Ubuntu) (64-bit and 32-bit)
- Mac OS X



## **Chapter 2**

# **General API Conventions**

## 2.1 Memory Management

The design of all the NPP functions follows the same guidelines as other NVIDIA CUDA libraries like cuFFT and cuBLAS. That is that all pointer arguments in those APIs are device pointers.

This convention enables the individual developer to make smart choices about memory management that minimize the number of memory transfers. It also allows the user the maximum flexibility regarding which of the various memory transfer mechanisms offered by the CUDA runtime is used, e.g. synchronous or asynchronous memory transfers, zero-copy and pinned memory, etc.

The most basic steps involved in using NPP for processing data is as follows:

1. Transfer input data from the host to device using

```
cudaMemcpy(...)
```

2. Process data using one or several NPP functions or custom CUDA kernels

3. Transfer the result data from the device to the host using

```
cudaMemcpy(...)
```

Throughout NPP there are a number of functions that require the use of host pointers. E.g. various `<Primitive>GetBufferSize(...)` functions. Those functions compute the minimum size of (scratch memory) buffer that some primitives require. This buffer size is returned via a host pointer. Since these buffers are allocated via CUDA runtime functions, it would make no sense to place those size values in device memory by default.

## 2.2 Function Naming

Since NPP is a C API and therefore does not allow for function overloading for different data-types the NPP naming convention addresses the need to differentiate between different flavors of the same algorithm or primitive function but for various data types. This disambiguation of different flavors of a primitive is done via a suffix containing data type and other disambiguating information.

In addition to the flavor suffix, all NPP functions are prefixed with by the letters "npp". Primitives belonging to NPP's image-processing module add the letter "i" to the npp prefix, i.e. are prefixed by "nppi". Similarly signal-processing primitives are prefixed with "npps".

The general naming scheme is:

```
npp<module info><PrimitiveName>_<data-type info>[_<additional flavor info>](<parameter list>)
```

The data-type information uses the same names as the [Basic NPP Data Types](#). For example the data-type information "8u" would imply that the primitive operates on [Npp8u](#) data.

If a primitive consumes different type data from what it produces, both types will be listed in the order of consumed to produced data type.

Details about the "additional flavor information" is provided for each of the NPP modules, since each problem domain uses different flavor information suffixes.



## **Chapter 3**

# **Signal-Processing Specific API Conventions**

## 3.1 Signal Data

Signal data is passed to and from NPPS primitives via a pointer to the signal's data type.

The general idea behind this fairly low-level way of passing signal data is ease-of-adoption into existing software projects:

- Passing the data pointer rather than a higher-level signal struct allows for easy adoption by not requiring a specific signal representation (that could include total signal size offset, or other additional information). This avoids awkward packing and unpacking of signal data from the host application to an NPP specific signal representation.

### 3.1.1 Parameter Names for Signal Data

There are three general cases of image-data passing throughout NPP detailed in the following sections.

Those are signals consumed by the algorithm.

#### 3.1.1.1 Source Signal Pointer

The source signal data is generally passed via a pointer named

`pSrc`

The source signal pointer is generally defined constant, enforcing that the primitive does not change any image data pointed to by that pointer. E.g.

```
nppsPrimitive_32s(const Npp32s * pSrc, ...)
```

In case the primitive consumes multiple signals as inputs the source pointers are numbered like this:

`pSrc1, pSrc2, ...`

#### 3.1.1.2 Destination Signal Pointer

The destination signal data is generally passed via a pointer named

`pDst`

In case the primitive consumes multiple signals as inputs the source pointers are numbered like this:

`pDst1, pDst2, ...`

#### 3.1.1.3 In-Place Signal Pointer

In the case of in-place processing, source and destination are served by the same pointer and thus pointers to in-place signal data are called:

`pSrcDst`

### 3.1.2 Signal Data Alignment Requirements

NPP requires signal sample data to be naturally aligned, i.e. any pointer

```
NppType * p;
```

to a sample in a signal needs to fulfill:

```
assert(p % sizeof(p) == 0);
```

### 3.1.3 Signal Data Related Error Codes

All NPPI primitives operating on signal data validate the signal-data pointer for proper alignment and test that the point is not null.

Failed validation results in one of the following error codes being returned and the primitive not being executed:

- [NPP\\_NULL\\_POINTER\\_ERROR](#) is returned if the image-data pointer is 0 (NULL).
- [NPP\\_ALIGNMENT\\_ERROR](#) if the signal-data pointer address is not a multiple of the signal's data-type size.

## 3.2 Signal Length

The vast majority of NPPS functions take a

```
nLength
```

parameter that tells the primitive how many of the signal's samples starting from the given data pointer are to be processed.

### 3.2.1 Length Related Error Codes

All NPPS primitives taking a length parameter validate this input.

Failed validation results in the following error code being returned and the primitive not being executed:

- [NPP\\_SIZE\\_ERROR](#) is returned if the length is negative.



## **Chapter 4**

# **Imaging-Processing Specific API Conventions**

## 4.1 Function Naming

Image processing related functions use a number of suffixes to indicate various different flavors of a primitive beyond just different data types. The flavor suffix uses the following abbreviations:

- "A" if the image is a 4 channel image this indicates the result alpha channel is not affected by the primitive.
- "Cn" the image consists of n channel packed pixels, where n can be 1, 2, 3 or 4.
- "Pn" the image consists of n separate image planes, where n can be 1, 2, 3 or 4.
- "C" (following the channel information) indicates that the primitive only operates on one of the color channels, the "channel-of-interest". All other output channels are not affected by the primitive.
- "I" indicates that the primitive works "in-place". In this case the image-data pointer is usually named "pSrcDst" to indicate that the image data serves as source and destination at the same time.
- "M" indicates "masked operation". These types of primitives have an additional "mask image" as input. Each pixel in the destination image corresponds to a pixel in the mask image. Only pixels with a corresponding non-zero mask pixel are being processed.
- "R" indicates the primitive operates only on a rectangular "region-of-interest" or "ROI". All ROI primitives take an additional input parameter of type [NppiSize](#), which specifies the width and height of the rectangular region that the primitive should process. For details on how primitives operate on ROIs see: [Region-of-Interest \(ROI\)](#).
- "Sfs" indicates the result values are processed by fixed scaling and saturation before they're written out.

The suffixes above always appear in alphabetical order. E.g. a 4 channel primitive not affecting the alpha channel with masked operation, in place and with scaling/saturation and ROI would have the postfix: "AC4IMRSfs".

## 4.2 Image Data

Image data is passed to and from NPPI primitives via a pair of parameters:

1. A pointer to the image's underlying data type.
2. A line step in bytes (also sometimes called line stride).

The general idea behind this fairly low-level way of passing image data is ease-of-adoption into existing software projects:

- Passing a raw pointer to the underlying pixel data type, rather than structured (by color) channel pixel data allows usage of the function in a wide variety of situations avoiding risky type cast or expensive image data copies.
- Passing the data pointer and line step individually rather than a higher-level image struct again allows for easy adoption by not requiring a specific image representation and thus avoiding awkward packing and unpacking of image data from the host application to an NPP specific image representation.

### 4.2.1 Line Step

The line step (also called "line stride" or "row step") allows lines of oddly sized images to start on well-aligned addresses by adding a number of unused bytes at the ends of the lines. This type of line padding has been common practice in digital image processing for a long time and is not particular to GPU image processing.

The line step is the number of bytes in a line **including the padding**. An other way to interpret this number is to say that it is the number of bytes between the first pixel of successive rows in the image, or generally the number of bytes between two neighboring pixels in any column of pixels.

The general reason for the existence of the line step it is that uniformly aligned rows of pixel enable optimizations of memory-access patterns.

Even though all functions in NPP will work with arbitrarily aligned images, best performance can only be achieved with well aligned image data. Any image data allocated with the NPP image allocators or the 2D memory allocators in the CUDA runtime, is well aligned.

Particularly on older CUDA capable GPUs it is likely that the performance decrease for misaligned data is substantial (orders of magnitude).

All image data passed to NPPI primitives requires a line step to be provided. It is important to keep in mind that this line step is always specified in terms of bytes, not pixels.

### 4.2.2 Parameter Names for Image Data

There are three general cases of image-data passing throughout NPP detailed in the following sections.

#### 4.2.2.1 Passing Source-Image Data

Those are images consumed by the algorithm.

##### 4.2.2.1.1 Source-Image Pointer

The source image data is generally passed via a pointer named

```
pSrc
```

The source image pointer is generally defined constant, enforcing that the primitive does not change any image data pointed to by that pointer. E.g.

```
nppiPrimitive_32s_C1R(const Npp32s * pSrc, ...)
```

In case the primitive consumes multiple images as inputs the source pointers are numbered like this:

```
pSrc1, pSrc2, ...
```

##### 4.2.2.1.2 Source-Image Line Step

The source-image line step is the number of bytes between successive rows in the image. The source-image line step parameter is

```
nSrcStep
```

or in the case of multiple source images

```
nSrcStep1, nSrcStep2, ...
```

#### 4.2.2.2 Passing Destination-Image Data

Those are images produced by the algorithm.

##### 4.2.2.2.1 Destination-Image Pointer

The destination image data is generally passed via a pointer named

```
pDst
```

In case the primitive consumes multiple images as inputs the source pointers are numbered like this:

```
pDst1, pDst2, ...
```

##### 4.2.2.2.2 Destination-Image Line Step

The destination-image line step parameter is

```
nDstStep
```

or in the case of multiple destination images

```
nDstStep1, nDstStep2, ...
```

#### 4.2.2.3 Passing In-Place Image Data

##### 4.2.2.3.1 In-Place Image Pointer

In the case of in-place processing, source and destination are served by the same pointer and thus pointers to in-place image data are called:

```
pSrcDst
```

##### 4.2.2.3.2 In-Place Line Step

The in-place nSrcDstStep

### 4.2.3 Image Data Alignment Requirements

NPP requires pixel data to adhere to certain alignment constraints: For 2 and 4 channel images the following alignment requirement holds: `data_pointer % (#channels * sizeof(channel type)) == 0`. E.g. a 4 channel image with underlying type [Npp8u](#) (8-bit unsigned) would require all pixels to fall on addresses that are multiples of 4 (4 channels \* 1 byte size).

As a logical consequence of all pixels being aligned to their natural size the image line steps of 2 and 4 channel images also need to be multiples of the pixel size.

1 and 3 channel images only require that pixel pointers are aligned to the underlying data type, i.e. `pData % sizeof(data type) == 0`. And consequentially line steps are also held to this requirement.



#### 4.2.4 Image Data Related Error Codes

All NPPI primitives operating on image data validate the image-data pointer for proper alignment and test that the point is not null. They also validate the line stride for proper alignment and guard against the step being less or equal to 0. Failed validation results in one of the following error codes being returned and the primitive not being executed:

- [NPP\\_STEP\\_ERROR](#) is returned if the data step is 0 or negative.
- [NPP\\_NOT\\_EVEN\\_STEP\\_ERROR](#) is returned if the line step is not a multiple of the pixel size for 2 and 4 channel images.
- [NPP\\_NULL\\_POINTER\\_ERROR](#) is returned if the image-data pointer is 0 (NULL).
- [NPP\\_ALIGNMENT\\_ERROR](#) if the image-data pointer address is not a multiple of the pixel size for 2 and 4 channel images.

### 4.3 Region-of-Interest (ROI)

In practice processing a rectangular sub-region of an image is often more common than processing complete images. The vast majority of NPP's image-processing primitives allow for processing of such sub regions also referred to as regions-of-interest or ROIs.

All primitives supporting ROI processing are marked by a "R" in their name suffix. Where possible, the ROI a primitive operates on is passed as a single [NppiSize](#) struct, which provides the width and height of the ROI. This raises the obvious question how the primitive knows where in the image this rectangle of (width, height) is located. The "start pixel" of the ROI is implicitly given by the image-data pointer. I.e. instead of explicitly passing a pixel coordinate for the upper-right corner of the ROI the primitive's user needs to perform the necessary offset computation on the image data pointers, such that the pointers passed to the primitive thus point to the start of the ROI.

In practice this means that for an image (pSrc, nSrcStep) and the start-pixel of the ROI being given by (xROI, yROI), one would pass

$pSrcOffset = pSrc + yROI * nSrcStep + xROI * PixelSize;$

as the image-data source to the primitive. PixelSize is typically computed as

$PixelSize = NumberOfColorChannels * sizeof(PixelDataType).$

E.g. for a primitive like [nppiSet\\_16s\\_C4R\(\)](#) we would have

- $NumberOfColorChannels == 4;$
- $sizeof(Npp16s) == 2;$
- and thus  $PixelSize = 4 * 2 = 8;$

#### 4.3.1 ROI Related Error Codes

All NPPI primitives operating on ROIs of image data validate the ROI size and image's step size. Failed validation results in one of the following error codes being returned and the primitive not being executed:

- [NPP\\_SIZE\\_ERROR](#) is returned if either the ROI width or ROI height are negative.
- [NPP\\_STEP\\_ERROR](#) is returned if the ROI width exceeds the image's line step. In mathematical terms  $(widthROI * PixelSize) > nLinStep$  indicates an error.



# Chapter 5

## Module Index

### 5.1 Modules

Here is a list of all modules:

NPP Core . . . . .	19
NPP Type Definitions and Constants . . . . .	21
Basic NPP Data Types . . . . .	29
NPP Image Processing . . . . .	31
NPP Signal Processing . . . . .	218



## Chapter 6

# Data Structure Index

### 6.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Npp16sc</a> (Complex Number This struct represents a short complex number ) . . . . .	251
<a href="#">Npp32fc</a> (Complex Number This struct represents a single floating-point complex number ) . . .	252
<a href="#">Npp32sc</a> (Complex Number This struct represents a signed int complex number ) . . . . .	253
<a href="#">Npp64fc</a> (Complex Number This struct represents a double floating-point complex number ) . .	254
<a href="#">Npp64sc</a> (Complex Number This struct represents a long long complex number ) . . . . .	255
<a href="#">NppiHaarBuffer</a> . . . . .	256
<a href="#">NppiHaarClassifier_32f</a> . . . . .	257
<a href="#">NppiPoint</a> (2D Point ) . . . . .	258
<a href="#">NppiRect</a> (2D Rectangle This struct contains position and size information of a rectangle in two space ) . . . . .	259
<a href="#">NppiSize</a> (2D Size This struct typically represents the size of a a rectangular region in two space )	260
<a href="#">NppLibraryVersion</a> . . . . .	261



# Chapter 7

## Module Documentation

### 7.1 NPP Core

Basic functions for library management, in particular library version and device property query functions.

#### Functions

- `const NppLibraryVersion * nppGetLibVersion ()`  
*Get the NPP library version.*
- `NppGpuComputeCapability nppGetGpuComputeCapability ()`  
*What CUDA compute model is supported by the default CUDA device?*
- `int nppGetGpuNumSMs ()`  
*Get the number of Streaming Multiprocessors (SM) on the default CUDA device.*
- `int nppGetMaxThreadsPerBlock ()`  
*Get the maximum number of threads per block on the default CUDA device.*
- `const char * nppGetGpuName ()`  
*Get the name of the default CUDA device.*

#### 7.1.1 Detailed Description

Basic functions for library management, in particular library version and device property query functions.

#### 7.1.2 Function Documentation

##### 7.1.2.1 NppGpuComputeCapability nppGetGpuComputeCapability ()

What CUDA compute model is supported by the default CUDA device?

Before trying to call any NPP functions, the user should make a call this function to ensure that the current machine has a CUDA capable device.

**Returns:**

An enum value representing if a CUDA capable device was found and what level of compute capabilities it supports.

**7.1.2.2 `const char* nppGetGpuName ()`**

Get the name of the default CUDA device.

**Returns:**

Name string of the graphics-card/compute device in a system.

**7.1.2.3 `int nppGetGpuNumSMs ()`**

Get the number of Streaming Multiprocessors (SM) on the default CUDA device.

**Returns:**

Number of SMs of the default CUDA device.

**7.1.2.4 `const NppLibraryVersion* nppGetLibVersion ()`**

Get the NPP library version.

**Returns:**

A struct containing separate values for major and minor revision and build number.

**7.1.2.5 `int nppGetMaxThreadsPerBlock ()`**

Get the maximum number of threads per block on the default CUDA device.

**Returns:**

Maximum number of threads per block on the default CUDA device.



## 7.2 NPP Type Definitions and Constants

### Data Structures

- struct [NppLibraryVersion](#)
- struct [NppiPoint](#)  
*2D Point*
- struct [NppiSize](#)  
*2D Size This struct typically represents the size of a rectangular region in two space.*
- struct [NppiRect](#)  
*2D Rectangle This struct contains position and size information of a rectangle in two space.*
- struct [NppiHaarClassifier\\_32f](#)
- struct [NppiHaarBuffer](#)

### Modules

- [Basic NPP Data Types](#)

### Defines

- #define [NPP\\_MIN\\_8U](#) ( 0 )  
*Minimum 8-bit unsigned integer.*
- #define [NPP\\_MAX\\_8U](#) ( 255 )  
*Maximum 8-bit unsigned integer.*
- #define [NPP\\_MIN\\_16U](#) ( 0 )  
*Minimum 16-bit unsigned integer.*
- #define [NPP\\_MAX\\_16U](#) ( 65535 )  
*Maximum 16-bit unsigned integer.*
- #define [NPP\\_MIN\\_32U](#) ( 0 )  
*Minimum 32-bit unsigned integer.*
- #define [NPP\\_MAX\\_32U](#) ( 4294967295 )  
*Maximum 32-bit unsigned integer.*
- #define [NPP\\_MIN\\_8S](#) (-128 )  
*Minimum 8-bit signed integer.*
- #define [NPP\\_MAX\\_8S](#) ( 127 )  
*Maximum 8-bit signed integer.*
- #define [NPP\\_MIN\\_16S](#) (-32768 )  
*Minimum 16-bit signed integer.*

- #define `NPP_MAX_16S` ( 32767 )  
*Maximum 16-bit signed integer.*
- #define `NPP_MIN_32S` (-2147483648 )  
*Minimum 32-bit signed integer.*
- #define `NPP_MAX_32S` ( 2147483647 )  
*Maximum 32-bit signed integer.*
- #define `NPP_MAX_64S` ( 9223372036854775807LL )  
*Minimum 64-bit signed integer.*
- #define `NPP_MIN_64S` (-9223372036854775808LL)  
*Maximum 64-bit signed integer.*
- #define `NPP_MINABS_32F` ( 1.175494351e-38f )  
*Smallest positive 32-bit floating point value.*
- #define `NPP_MAXABS_32F` ( 3.402823466e+38f )  
*Largest positive 32-bit floating point value.*
- #define `NPP_MINABS_64F` ( 2.2250738585072014e-308 )  
*Smallest positive 64-bit floating point value.*
- #define `NPP_MAXABS_64F` ( 1.7976931348623158e+308 )  
*Largest positive 64-bit floating point value.*

## Enumerations

- enum `NppiInterpolationMode` {  
`NPPI_INTER_NN` = 1,  
`NPPI_INTER_LINEAR` = 2,  
`NPPI_INTER_CUBIC` = 4,  
`NPPI_INTER_SUPER` = 8,  
`NPPI_INTER_LANCZOS` = 16,  
`NPPI_SMOOTH_EDGE` = (1 << 31) }  
*Filtering methods.*
- enum `NppStatus` {  
`NPP_NOT_SUPPORTED_MODE_ERROR` = -9999,  
`NPP_ROUND_MODE_NOT_SUPPORTED_ERROR` = -213,  
`NPP_RESIZE_NO_OPERATION_ERROR` = -50,  
`NPP_NOT_SUFFICIENT_COMPUTE_CAPABILITY` = -27,  
`NPP_BAD_ARG_ERROR` = -26,  
`NPP_LUT_NUMBER_OF_LEVELS_ERROR` = -25,

```

NPP_TEXTURE_BIND_ERROR = -24,
NPP_COEFF_ERROR = -23,
NPP_RECT_ERROR = -22,
NPP_QUAD_ERROR = -21,
NPP_WRONG_INTERSECTION_ROI_ERROR = -20,
NPP_NOT_EVEN_STEP_ERROR = -19,
NPP_INTERPOLATION_ERROR = -18,
NPP_RESIZE_FACTOR_ERROR = -17,
NPP_HAAR_CLASSIFIER_PIXEL_MATCH_ERROR = -16,
NPP_MEMFREE_ERR = -15,
NPP_MEMSET_ERR = -14,
NPP_MEMCPY_ERROR = -13,
NPP_MEM_ALLOC_ERR = -12,
NPP_HISTO_NUMBER_OF_LEVELS_ERROR = -11,
NPP_MIRROR_FLIP_ERR = -10,
NPP_INVALID_INPUT = -9,
NPP_ALIGNMENT_ERROR = -8,
NPP_STEP_ERROR = -7,
NPP_SIZE_ERROR = -6,
NPP_POINTER_ERROR = -5,
NPP_NULL_POINTER_ERROR = -4,
NPP_CUDA_KERNEL_EXECUTION_ERROR = -3,
NPP_NOT_IMPLEMENTED_ERROR = -2,
NPP_ERROR = -1,
NPP_NO_ERROR = 0,
NPP_SUCCESS = NPP_NO_ERROR,
NPP_WARNING = 1,
NPP_WRONG_INTERSECTION_QUAD_WARNING = 2,
NPP_MISALIGNED_DST_ROI_WARNING = 3,
NPP_AFFINE_QUAD_INCORRECT_WARNING = 4,
NPP_DOUBLE_SIZE_WARNING = 5,
NPP_ODD_ROI_WARNING = 6 }

```

*Error Status Codes.*

- enum NppGpuComputeCapability {
 

```

NPP_CUDA_UNKNOWN_VERSION = -1,
NPP_CUDA_NOT_CAPABLE,
NPP_CUDA_1_0,
NPP_CUDA_1_1,
NPP_CUDA_1_2,
NPP_CUDA_1_3,
NPP_CUDA_2_0 }

```

- enum `NppiAxis` {  
`NPP_HORIZONTAL_AXIS`,  
`NPP_VERTICAL_AXIS`,  
`NPP_BOTH_AXIS` }
- enum `NppCmpOp` {  
`NPP_CMP_LESS`,  
`NPP_CMP_LESS_EQ`,  
`NPP_CMP_EQ`,  
`NPP_CMP_GREATER_EQ`,  
`NPP_CMP_GREATER` }
- enum `NppRoundMode` {  
`NPP_RND_ZERO`,  
`NPP_RND_NEAR`,  
`NPP_RND_FINANCIAL` }
- enum `NppHintAlgorithm` {  
`nppAlgHintNone`,  
`nppAlgHintFast`,  
`nppAlgHintAccurate` }

## 7.2.1 Define Documentation

### 7.2.1.1 `#define NPP_MAX_16S ( 32767 )`

Maximum 16-bit signed integer.

### 7.2.1.2 `#define NPP_MAX_16U ( 65535 )`

Maximum 16-bit unsigned integer.

### 7.2.1.3 `#define NPP_MAX_32S ( 2147483647 )`

Maximum 32-bit signed integer.

### 7.2.1.4 `#define NPP_MAX_32U ( 4294967295 )`

Maximum 32-bit unsigned integer.

### 7.2.1.5 `#define NPP_MAX_64S ( 9223372036854775807LL )`

Minimum 64-bit signed integer.

### 7.2.1.6 `#define NPP_MAX_8S ( 127 )`

Maximum 8-bit signed integer.

**7.2.1.7 #define NPP\_MAX\_8U ( 255 )**

Maximum 8-bit unsigned integer.

**7.2.1.8 #define NPP\_MAXABS\_32F ( 3.402823466e+38f )**

Largest positive 32-bit floating point value.

**7.2.1.9 #define NPP\_MAXABS\_64F ( 1.7976931348623158e+308 )**

Largest positive 64-bit floating point value.

**7.2.1.10 #define NPP\_MIN\_16S (-32768 )**

Minimum 16-bit signed integer.

**7.2.1.11 #define NPP\_MIN\_16U ( 0 )**

Minimum 16-bit unsigned integer.

**7.2.1.12 #define NPP\_MIN\_32S (-2147483648 )**

Minimum 32-bit signed integer.

**7.2.1.13 #define NPP\_MIN\_32U ( 0 )**

Minimum 32-bit unsigned integer.

**7.2.1.14 #define NPP\_MIN\_64S (-9223372036854775808LL)**

Maximum 64-bit signed integer.

**7.2.1.15 #define NPP\_MIN\_8S (-128 )**

Minimum 8-bit signed integer.

**7.2.1.16 #define NPP\_MIN\_8U ( 0 )**

Minimum 8-bit unsigned integer.

**7.2.1.17 #define NPP\_MINABS\_32F ( 1.175494351e-38f )**

Smallest positive 32-bit floating point value.

### 7.2.1.18 #define NPP\_MINABS\_64F ( 2.2250738585072014e-308 )

Smallest positive 64-bit floating point value.

## 7.2.2 Enumeration Type Documentation

### 7.2.2.1 enum NppCmpOp

Enumerator:

*NPP\_CMP\_LESS*  
*NPP\_CMP\_LESS\_EQ*  
*NPP\_CMP\_EQ*  
*NPP\_CMP\_GREATER\_EQ*  
*NPP\_CMP\_GREATER*

### 7.2.2.2 enum NppGpuComputeCapability

Enumerator:

*NPP\_CUDA\_UNKNOWN\_VERSION* Indicates that the compute-capability query failed.  
*NPP\_CUDA\_NOT\_CAPABLE* Indicates that no CUDA capable device was found on machine.  
*NPP\_CUDA\_1\_0* Indicates that CUDA 1.0 capable device is default device on machine.  
*NPP\_CUDA\_1\_1* Indicates that CUDA 1.1 capable device.  
*NPP\_CUDA\_1\_2* Indicates that CUDA 1.2 capable device.  
*NPP\_CUDA\_1\_3* Indicates that CUDA 1.3 capable device.  
*NPP\_CUDA\_2\_0* Indicates that CUDA 2.0 or better is default device on machine.

### 7.2.2.3 enum NppHintAlgorithm

Enumerator:

*nppAlgHintNone*  
*nppAlgHintFast*  
*nppAlgHintAccurate*

### 7.2.2.4 enum NppiAxis

Enumerator:

*NPP\_HORIZONTAL\_AXIS*  
*NPP\_VERTICAL\_AXIS*  
*NPP\_BOTH\_AXIS*

### 7.2.2.5 enum NppiInterpolationMode

Filtering methods.

**Enumerator:**

*NPPI\_INTER\_NN* Nearest neighbor filtering.  
*NPPI\_INTER\_LINEAR* Linear interpolation.  
*NPPI\_INTER\_CUBIC* Cubic interpolation.  
*NPPI\_INTER\_SUPER* Super sampling.  
*NPPI\_INTER\_LANCZOS* Lanczos filtering.  
*NPPI\_SMOOTH\_EDGE* Smooth edge filtering.

### 7.2.2.6 enum NppRoundMode

**Enumerator:**

*NPP\_RND\_ZERO*  
*NPP\_RND\_NEAR*  
*NPP\_RND\_FINANCIAL*

### 7.2.2.7 enum NppStatus

Error Status Codes.

Almost all NPP function return error-status information using these return codes. Negative return codes indicate errors, positive return codes indicate warnings, a return code of 0 indicates success.

**Enumerator:**

*NPP\_NOT\_SUPPORTED\_MODE\_ERROR*  
*NPP\_ROUND\_MODE\_NOT\_SUPPORTED\_ERROR*  
*NPP\_RESIZE\_NO\_OPERATION\_ERROR*  
*NPP\_NOT\_SUFFICIENT\_COMPUTE\_CAPABILITY*  
*NPP\_BAD\_ARG\_ERROR*  
*NPP\_LUT\_NUMBER\_OF\_LEVELS\_ERROR*  
*NPP\_TEXTURE\_BIND\_ERROR*  
*NPP\_COEFF\_ERROR*  
*NPP\_RECT\_ERROR*  
*NPP\_QUAD\_ERROR*  
*NPP\_WRONG\_INTERSECTION\_ROI\_ERROR*  
*NPP\_NOT\_EVEN\_STEP\_ERROR*  
*NPP\_INTERPOLATION\_ERROR*  
*NPP\_RESIZE\_FACTOR\_ERROR*  
*NPP\_HAAR\_CLASSIFIER\_PIXEL\_MATCH\_ERROR*  
*NPP\_MEMFREE\_ERR*

***NPP\_MEMSET\_ERR***

***NPP\_MEMCPY\_ERROR***

***NPP\_MEM\_ALLOC\_ERR***

***NPP\_HISTO\_NUMBER\_OF\_LEVELS\_ERROR***

***NPP\_MIRROR\_FLIP\_ERR***

***NPP\_INVALID\_INPUT***

***NPP\_ALIGNMENT\_ERROR***

***NPP\_STEP\_ERROR*** Step is less or equal zero.

***NPP\_SIZE\_ERROR***

***NPP\_POINTER\_ERROR***

***NPP\_NULL\_POINTER\_ERROR***

***NPP\_CUDA\_KERNEL\_EXECUTION\_ERROR***

***NPP\_NOT\_IMPLEMENTED\_ERROR***

***NPP\_ERROR***

***NPP\_NO\_ERROR*** Error free operation.

***NPP\_SUCCESS*** Successful operation (same as NPP\_NO\_ERROR).

***NPP\_WARNING***

***NPP\_WRONG\_INTERSECTION\_QUAD\_WARNING***

***NPP\_MISALIGNED\_DST\_ROI\_WARNING*** Speed reduction due to uncoalesced memory accesses warning.

***NPP\_AFFINE\_QUAD\_INCORRECT\_WARNING*** Indicates that the quadrangle passed to one of affine warping functions doesn't have necessary properties. First 3 vertices are used, the fourth vertex discarded.

***NPP\_DOUBLE\_SIZE\_WARNING*** Indicates that in case of 422/411/420 sampling the ROI width/height was modified for proper processing.

***NPP\_ODD\_ROI\_WARNING*** Indicates that for 422/411/420 sampling the ROI width/height was forced to even value.



## 7.3 Basic NPP Data Types

### Data Structures

- struct [Npp16sc](#)  
*Complex Number This struct represents a short complex number.*
- struct [Npp32sc](#)  
*Complex Number This struct represents a signed int complex number.*
- struct [Npp32fc](#)  
*Complex Number This struct represents a single floating-point complex number.*
- struct [Npp64sc](#)  
*Complex Number This struct represents a long long complex number.*
- struct [Npp64fc](#)  
*Complex Number This struct represents a double floating-point complex number.*

### Typedefs

- typedef unsigned char [Npp8u](#)  
*8-bit unsigned chars*
- typedef signed char [Npp8s](#)  
*8-bit signed chars*
- typedef unsigned short [Npp16u](#)  
*16-bit unsigned integers*
- typedef short [Npp16s](#)  
*16-bit signed integers*
- typedef unsigned int [Npp32u](#)  
*32-bit unsigned integers*
- typedef int [Npp32s](#)  
*32-bit signed integers*
- typedef unsigned long long [Npp64u](#)  
*64-bit unsigned integers*
- typedef long long [Npp64s](#)  
*64-bit signed integers*
- typedef float [Npp32f](#)  
*32-bit (IEEE) floating-point numbers*

- typedef double [Npp64f](#)  
*64-bit floating-point numbers*

### 7.3.1 Typedef Documentation

#### 7.3.1.1 typedef short Npp16s

16-bit signed integers

#### 7.3.1.2 typedef unsigned short Npp16u

16-bit unsigned integers

#### 7.3.1.3 typedef float Npp32f

32-bit (IEEE) floating-point numbers

#### 7.3.1.4 typedef int Npp32s

32-bit signed integers

#### 7.3.1.5 typedef unsigned int Npp32u

32-bit unsigned integers

#### 7.3.1.6 typedef double Npp64f

64-bit floating-point numbers

#### 7.3.1.7 typedef long long Npp64s

64-bit signed integers

#### 7.3.1.8 typedef unsigned long long Npp64u

64-bit unsigned integers

#### 7.3.1.9 typedef signed char Npp8s

8-bit signed chars

#### 7.3.1.10 typedef unsigned char Npp8u

8-bit unsigned chars

## 7.4 NPP Image Processing

### Functions

- `NppStatus nppiSqrIntegral_8u32s32f_C1R` (`Npp8u *pSrc`, `int nSrcStep`, `Npp32s *pDst`, `int nDstStep`, `Npp32f *pSqr`, `int nSqrStep`, `NppiSize srcROI`, `Npp32s val`, `Npp32f valSqr`, `Npp32s integral-ImageNewHeight`)

*SqrIntegral Transforms an image to integral and integral of pixel squares representation.*

- `NppStatus nppiRectStdDev_32s32f_C1R` (`const Npp32s *pSrc`, `int nSrcStep`, `const Npp32f *pSqr`, `int nSqrStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `NppiRect rect`)

*RectStdDev Computes the standard deviation of integral images.*

### Image-Memory Allocation

ImageAllocator methods for 2D arrays of data.

The allocators have width and height parameters to specify the size of the image data being allocated. They return a pointer to the newly created memory and return the numbers of bytes between successive lines.

All allocators return memory with line strides that are beneficial for performance. It is not mandatory to use these allocators. Any valid CUDA device-memory pointers can be used by the NPP primitives and there are no restrictions on line strides.

- `Npp8u * nppiMalloc_8u_C1` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*8-bit unsigned image memory allocator.*
- `Npp8u * nppiMalloc_8u_C2` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*2 channel 8-bit unsigned image memory allocator.*
- `Npp8u * nppiMalloc_8u_C3` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*3 channel 8-bit unsigned image memory allocator.*
- `Npp8u * nppiMalloc_8u_C4` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*4 channel 8-bit unsigned image memory allocator.*
- `Npp16u * nppiMalloc_16u_C1` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*16-bit unsigned image memory allocator.*
- `Npp16u * nppiMalloc_16u_C3` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*3 channel 16-bit unsigned image memory allocator.*
- `Npp16u * nppiMalloc_16u_C4` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*4 channel 16-bit unsigned image memory allocator.*
- `Npp16s * nppiMalloc_16s_C1` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*16-bit signed image memory allocator.*
- `Npp16s * nppiMalloc_16s_C4` (`int nWidthPixels`, `int nHeightPixels`, `int *pStepBytes`)  
*4 channel 16-bit signed image memory allocator.*

- `Npp32s * nppiMalloc_32s_C1` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*32-bit signed image memory allocator.*
- `Npp32s * nppiMalloc_32s_C3` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*3 channel 32-bit signed image memory allocator.*
- `Npp32s * nppiMalloc_32s_C4` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*4 channel 32-bit signed image memory allocator.*
- `Npp32f * nppiMalloc_32f_C1` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*32-bit floating point image memory allocator.*
- `Npp32f * nppiMalloc_32f_C2` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*2 channel 32-bit floating point image memory allocator.*
- `Npp32f * nppiMalloc_32f_C3` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*3 channel 32-bit floating point image memory allocator.*
- `Npp32f * nppiMalloc_32f_C4` (int nWidthPixels, int nHeightPixels, int \*pStepBytes)  
*4 channel 32-bit floating point image memory allocator.*
- `void nppiFree` (void \*pData)  
*Free method for any 2D allocated memory.*

## Image-Memory Set

Set methods for images of various types.

Images are passed to these primitives via a pointer to the image data (first pixel in the ROI) and a step-width, i.e. the number of bytes between successive lines. The size of the area to be set (region-of-interest, ROI) is specified via a `NppiSize` struct. In addition to the image data and ROI, all methods have a parameter to specify the value being set. In case of single channel images this is a single value, in case of multi-channel, an array of values is passed.

- `NppStatus nppiSet_8u_C1R` (Npp8u nValue, Npp8u \*pDst, int nDstStep, NppiSize oSizeROI)  
*8-bit unsigned image set.*
- `NppStatus nppiSet_8u_C1MR` (Npp8u nValue, Npp8u \*pDst, int nDstStep, NppiSize oSizeROI, const Npp8u \*pMask, int nMaskStep)  
*Masked 8-bit unsigned image set.*
- `NppStatus nppiSet_8u_C4R` (const Npp8u aValues[4], Npp8u \*pDst, int nDstStep, NppiSize oSizeROI)  
*4 channel 8-bit unsigned image set.*
- `NppStatus nppiSet_8u_C4MR` (const Npp8u aValues[4], Npp8u \*pDst, int nDstStep, NppiSize oSizeROI, const Npp8u \*pMask, int nMaskStep)  
*Masked 4 channel 8-bit unsigned image set.*

- `NppStatus nppiSet_8u_AC4R` (const `Npp8u` aValues[3], `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned image set method, not affecting Alpha channel.*
- `NppStatus nppiSet_8u_AC4MR` (const `Npp8u` aValues[3], `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, int nMaskStep)  
*Masked 4 channel 8-bit unsigned image set method, not affecting Alpha channel.*
- `NppStatus nppiSet_8u_C4CR` (`Npp8u` nValue, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned image set affecting only single channel.*
- `NppStatus nppiSet_16u_C1R` (`Npp16u` nValue, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit unsigned image set.*
- `NppStatus nppiSet_16u_C1MR` (`Npp16u` nValue, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, int nMaskStep)  
*Masked 16-bit unsigned image set.*
- `NppStatus nppiSet_16u_C2R` (const `Npp16u` aValues[2], `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*2 channel 16-bit unsigned image set.*
- `NppStatus nppiSet_16u_C4R` (const `Npp16u` aValues[4], `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit unsigned image set.*
- `NppStatus nppiSet_16u_C4MR` (const `Npp16u` aValues[4], `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, int nMaskStep)  
*Masked 4 channel 16-bit unsigned image set.*
- `NppStatus nppiSet_16u_AC4R` (const `Npp16u` aValues[3], `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit unsigned image set method, not affecting Alpha channel.*
- `NppStatus nppiSet_16u_AC4MR` (const `Npp16u` aValues[3], `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, int nMaskStep)  
*Masked 4 channel 16-bit unsigned image set method, not affecting Alpha channel.*
- `NppStatus nppiSet_16u_C4CR` (`Npp16u` nValue, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit unsigned image set affecting only single channel.*
- `NppStatus nppiSet_16s_C1R` (`Npp16s` nValue, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit image set.*
- `NppStatus nppiSet_16s_C1MR` (`Npp16s` nValue, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, int nMaskStep)  
*Masked 16-bit image set.*

- **NppStatus nppiSet\_16s\_C2R** (const **Npp16s** aValues[2], **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*2 channel 16-bit image set.*
- **NppStatus nppiSet\_16s\_C4R** (const **Npp16s** aValues[4], **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 16-bit image set.*
- **NppStatus nppiSet\_16s\_C4MR** (const **Npp16s** aValues[4], **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 16-bit image set.*
- **NppStatus nppiSet\_16s\_AC4R** (const **Npp16s** aValues[3], **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 16-bit image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_16s\_AC4MR** (const **Npp16s** aValues[3], **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 16-bit image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_16s\_C4CR** (**Npp16s** nValue, **Npp16s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 16-bit unsigned image set affecting only single channel.*
- **NppStatus nppiSet\_32s\_C1R** (**Npp32s** nValue, **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*32-bit image set.*
- **NppStatus nppiSet\_32s\_C1MR** (**Npp32s** nValue, **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 32-bit image set.*
- **NppStatus nppiSet\_32s\_C4R** (const **Npp32s** aValues[4], **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 32-bit image set.*
- **NppStatus nppiSet\_32s\_C4MR** (const **Npp32s** aValues[4], **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 32-bit image set.*
- **NppStatus nppiSet\_32s\_AC4R** (const **Npp32s** aValues[3], **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 16-bit image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_32s\_AC4MR** (const **Npp32s** aValues[3], **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 16-bit image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_32s\_C4CR** (**Npp32s** nValue, **Npp32s** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 32-bit unsigned image set affecting only single channel.*
- **NppStatus nppiSet\_32f\_C1R** (**Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*32-bit floating point image set.*

- **NppStatus nppiSet\_32f\_C1MR** (**Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 32-bit floating point image set.*
- **NppStatus nppiSet\_32f\_C4R** (const **Npp32f** aValues[4], **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 32-bit floating point image set.*
- **NppStatus nppiSet\_32f\_C4MR** (const **Npp32f** aValues[4], **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 32-bit floating point image set.*
- **NppStatus nppiSet\_32f\_AC4R** (const **Npp32f** aValues[3], **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 32-bit floating point image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_32f\_AC4MR** (const **Npp32f** aValues[3], **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** \*pMask, int nMaskStep)  
*Masked 4 channel 32-bit floating point image set method, not affecting Alpha channel.*
- **NppStatus nppiSet\_32f\_C4CR** (**Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 32-bit floating point image set affecting only single channel.*

## Image-Memory Copy

Copy methods for images of various types.

Images are passed to these primitives via a pointer to the image data (first pixel in the ROI) and a step-width, i.e. the number of bytes between successive lines. The size of the area to be copied (region-of-interest, ROI) is specified via a **NppiSize** struct.

- **NppStatus nppiCopy\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*8-bit unsigned image copy.*
- **NppStatus nppiCopy\_8u\_C4R** (const **Npp8u** \*pSrc, int nSrcStep, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 8-bit unsigned image copy.*
- **NppStatus nppiCopy\_8u\_AC4R** (const **Npp8u** \*pSrc, int nSrcStep, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*4 channel 8-bit unsigned image copy, not affecting Alpha channel.*
- **NppStatus nppiCopy\_16u\_C1R** (const **Npp16u** \*pSrc, int nSrcStep, **Npp16u** \*pDst, int nDstStep, **NppiSize** oSizeROI)  
*16-bit unsigned image copy.*
- **NppStatus nppiCopy\_16u\_C4R** (const **Npp16u** \*pSrc, int nSrcStep, **Npp16u** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*4 channel 16-bit unsigned image copy.*

- `NppStatus nppiCopy_16u_AC4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 16-bit unsigned image copy, not affecting Alpha channel.*

- `NppStatus nppiCopy_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*16-bit image copy.*

- `NppStatus nppiCopy_16s_C4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 16-bit image copy.*

- `NppStatus nppiCopy_16s_AC4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 16-bit image copy, not affecting Alpha.*

- `NppStatus nppiCopy_32s_C1R` (const `Npp32s` \*pSrc, int nSrcStep, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit image copy.*

- `NppStatus nppiCopy_32s_C4R` (const `Npp32s` \*pSrc, int nSrcStep, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit image copy.*

- `NppStatus nppiCopy_32s_AC4R` (const `Npp32s` \*pSrc, int nSrcStep, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit image copy, not affecting Alpha.*

- `NppStatus nppiCopy_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point image copy.*

- `NppStatus nppiCopy_32f_C4R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit floating point image copy.*

- `NppStatus nppiCopy_32f_AC4R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit floating point image copy, not affecting Alpha.*

## Bit-Depth Conversion

Convert bit-depth up and down.

The integer conversion methods do not involve any scaling. Conversions that reduce bit-depth saturate values exceeding the reduced range to the range's maximum/minimum value. When converting from floating-point values to integer values, a rounding mode can be specified. After rounding to integer values the values get saturated to the destination data type's range.



- `NppStatus nppiConvert_8u16u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*8-bit unsigned to 16-bit unsigned conversion.*
- `NppStatus nppiConvert_16u8u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit unsigned to 8-bit unsigned conversion.*
- `NppStatus nppiConvert_8u16u_C4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned to 16-bit unsigned conversion.*
- `NppStatus nppiConvert_16u8u_C4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit unsigned to 8-bit unsigned conversion.*
- `NppStatus nppiConvert_8u16u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned to 16-bit unsigned conversion, not affecting Alpha.*
- `NppStatus nppiConvert_16u8u_AC4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit unsigned to 8-bit unsigned conversion, not affecting Alpha.*
- `NppStatus nppiConvert_8u16s_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*8-bit unsigned to 16-bit signed conversion.*
- `NppStatus nppiConvert_16s8u_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit signed to 8-bit unsigned conversion.*
- `NppStatus nppiConvert_8u16s_C4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned to 16-bit signed conversion.*
- `NppStatus nppiConvert_16s8u_C4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit signed to 8-bit unsigned conversion, not affecting Alpha.*
- `NppStatus nppiConvert_8u16s_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned to 16-bit signed conversion, not affecting Alpha.*
- `NppStatus nppiConvert_16s8u_AC4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 16-bit signed to 8-bit unsigned conversion, not affecting Alpha.*
- `NppStatus nppiConvert_16s32f_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit signed to 32-bit floating point conversion.*

- `NppStatus nppiConvert_32f16s_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppRoundMode` eRoundMode)  
*32-bit floating point to 16-bit conversion.*
- `NppStatus nppiConvert_8u32f_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*8-bit unsigned to 32-bit floating point conversion.*
- `NppStatus nppiConvert_16u32f_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit unsigned to 32-bit floating point conversion.*
- `NppStatus nppiConvert_32f16u_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppRoundMode` eRoundMode)  
*32-bit floating point to 16-bit unsigned conversion.*
- `NppStatus nppiConvert_32f8u_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppRoundMode` eRoundMode)  
*32-bit floating point to 8-bit unsigned conversion.*
- `NppStatus nppiConvert_16u32s_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit unsigned to 32-bit signed conversion.*
- `NppStatus nppiConvert_16s32s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*16-bit to 32-bit conversion.*

## Copy Const Border

Methods for copying images and padding borders with a constant, user-specifiable color.

- `NppStatus nppiCopyConstBorder_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `NppiSize` oSrcSizeROI, `Npp8u` \*pDst, int nDstStep, `NppiSize` oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, `Npp8u` nValue)  
*8-bit unsigned image copy with constant border color.*
- `NppStatus nppiCopyConstBorder_8u_C4R` (const `Npp8u` \*pSrc, int nSrcStep, `NppiSize` oSrcSizeROI, `Npp8u` \*pDst, int nDstStep, `NppiSize` oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, const `Npp8u` aValue[4])  
*4-channel 8-bit unsigned image copy with constant border color.*
- `NppStatus nppiCopyConstBorder_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `NppiSize` oSrcSizeROI, `Npp8u` \*pDst, int nDstStep, `NppiSize` oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, const `Npp8u` aValue[3])  
*4 channel 8-bit unsigned image copy with constant border color.*

- **NppStatus nppiCopyConstBorder\_32s\_C1R** (const **Npp32s** \*pSrc, int nSrcStep, **NppiSize** oSrcSizeROI, **Npp32s** \*pDst, int nDstStep, **NppiSize** oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, **Npp32s** nValue)

*32-bit image copy with constant border color.*

## Image Transpose

Methods for transposing images of various types.

Like matrix transpose, image transpose is a mirror along the image's diagonal (upper-left to lower-right corner).

- **NppStatus nppiTranspose\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **Npp8u** \*pDst, int nDstStep, **NppiSize** oROI)

*8-bit image transpose.*

## Image Color Channel Swap

Methods for exchanging the color channels of an image.

The methods support arbitrary permutations of the original channels, including replication.

- **NppStatus nppiSwapChannels\_8u\_C4IR** (**Npp8u** \*pSrcDst, int nSrcDstStep, **NppiSize** oSizeROI, const int aDstOrder[4])

*4 channel 8-bit unsigned swap channels, in-place.*

## Arithmetic with Constant Values

Methods performing image arithmetic with the second operand being a constant rather than an image.

- **NppStatus nppiAddC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*32-bit floating point image add constant.*

- **NppStatus nppiSubC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*32-bit floating point image subtract constant.*

- **NppStatus nppiMulC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*32-bit floating point image multiply constant.*

- **NppStatus nppiDivC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** nValue, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*32-bit floating point image divide by constant.*

- **NppStatus nppiAbsDiffC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI, **Npp32f** nValue)

*32-bit floating point image absolute difference from constant.*

- `NppStatus nppiAddC_32fc_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32fc` nValue, `Npp32fc` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit complex floating point image add constant.*

- `NppStatus nppiSubC_32fc_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32fc` nValue, `Npp32fc` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit complex floating point image subtract constant.*

- `NppStatus nppiMulC_32fc_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32fc` nValue, `Npp32fc` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit complex floating point image multiply constant.*

- `NppStatus nppiDivC_32fc_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32fc` nValue, `Npp32fc` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit complex floating point image divide by constant.*

## Image Addition

Methods for adding two images.

- `NppStatus nppiAdd_8u_C1RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*8-bit unsigned image add.*

- `NppStatus nppiAdd_8u_C4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image add.*

- `NppStatus nppiAdd_8u_AC4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image add, not affecting Alpha.*

- `NppStatus nppiAdd_32f_C1R` (const `Npp32f` \*pSrc1, int nSrc1Step, const `Npp32f` \*pSrc2, int nSrc2Step, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point image add.*

- `NppStatus nppiAdd_32s_C1R` (const `Npp32s` \*pSrc1, int nSrc1Step, const `Npp32s` \*pSrc2, int nSrc2Step, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit image add.*

## Image Subtraction

Methods for subtracting one image from another.

- `NppStatus nppiSub_8u_C1RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*8-bit unsigned image subtraction.*

- `NppStatus nppiSub_8u_C4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image subtraction.*

- `NppStatus nppiSub_8u_AC4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image subtraction, not affecting Alpha.*

- `NppStatus nppiSub_32f_C1R` (const `Npp32f` \*pSrc1, int nSrc1Step, const `Npp32f` \*pSrc2, int nSrc2Step, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point image subtraction.*

- `NppStatus nppiSub_32s_C1R` (const `Npp32s` \*pSrc1, int nSrc1Step, const `Npp32s` \*pSrc2, int nSrc2Step, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit image subtraction.*

## Image Multiplication

Methods for multiplying two images.

- `NppStatus nppiMul_8u_C1RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*8-bit unsigned image multiplication.*

- `NppStatus nppiMul_8u_C4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image multiplication.*

- `NppStatus nppiMul_8u_AC4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image multiplication, not affecting Alpha.*

- `NppStatus nppiMul_32f_C1R` (const `Npp32f` \*pSrc1, int nSrc1Step, const `Npp32f` \*pSrc2, int nSrc2Step, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit floating point image multiplication.*

- `NppStatus nppiMul_32s_C1R` (const `Npp32s` \*pSrc1, int nSrc1Step, const `Npp32s` \*pSrc2, int nSrc2Step, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 32-bit image multiplication.*

## Image Division

Methods for dividing one image by another.

- `NppStatus nppiDiv_8u_C1RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*8-bit unsigned image division.*

- `NppStatus nppiDiv_8u_C4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image division.*

- `NppStatus nppiDiv_8u_AC4RSfs` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, int nScaleFactor)

*4 channel 8-bit unsigned image division, not affecting Alpha.*

- `NppStatus nppiDiv_32f_C1R` (const `Npp32f` \*pSrc1, int nSrc1Step, const `Npp32f` \*pSrc2, int nSrc2Step, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point image division.*

- `NppStatus nppiDiv_32s_C1R` (const `Npp32s` \*pSrc1, int nSrc1Step, const `Npp32s` \*pSrc2, int nSrc2Step, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit image division.*

## Image Absolute Difference Methods

Per-pixel absolute difference methods.

- `NppStatus nppiAbsDiff_8u_C1R` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*8-bit unsigned absolute difference.*

- `NppStatus nppiAbsDiff_8u_C4R` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 8-bit unsigned absolute difference.*

- `NppStatus nppiAbsDiff_8u_AC4R` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*4 channel 8-bit unsigned absolute difference, not affecting Alpha.*

- `NppStatus nppiAbsDiff_32f_C1R` (const `Npp32f` \*pSrc1, int nSrc1Step, const `Npp32f` \*pSrc2, int nSrc2Step, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point absolute difference.*

- `NppStatus nppiAbsDiff_32s_C1R` (const `Npp32s` \*pSrc1, int nSrc1Step, const `Npp32s` \*pSrc2, int nSrc2Step, `Npp32s` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit absolute difference.*

## Other Image Arithmetic

- `NppStatus nppiLn_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point logarithm.*

- **NppStatus nppiExp\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI)

*32-bit floating point exponentiation.*

## Image Threshold Methods

Threshold pixels.

- **NppStatus nppiThreshold\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **Npp32f** \*pDst, int nDstStep, **NppiSize** oSizeROI, **Npp32f** nThreshold, **NppCmpOp** eComparisonOperation)

*32-bit floating point threshold.*

- **NppStatus nppiThreshold\_8u\_AC4R** (const **Npp8u** \*pSrc, int nSrcStep, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, const **Npp8u** aThresholds[3], **NppCmpOp** eComparisonOperation)

*4 channel 8-bit unsigned image threshold, not affecting Alpha.*

## Image Compare Methods

Compare the pixels of two images and create a binary result image.

In case of multi-channel image types, the condition must be fulfilled for all channels, otherwise the comparison is considered false. The "binary" result image is of type 8u\_C1. False is represented by 0, true by NPP\_MAX\_8U.

- **NppStatus nppiCompare\_8u\_C4R** (const **Npp8u** \*pSrc1, int nSrc1Step, const **Npp8u** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)

*4 channel 8-bit unsigned image compare.*

- **NppStatus nppiCompare\_8u\_AC4R** (const **Npp8u** \*pSrc1, int nSrc1Step, const **Npp8u** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)

*4 channel 8-bit unsigned image compare, not affecting Alpha.*

- **NppStatus nppiCompare\_32f\_C1R** (const **Npp32f** \*pSrc1, int nSrc1Step, const **Npp32f** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)

*32-bit floating point image compare.*

## Mean\_StdDev

Computes the mean and standard deviation of image pixel values

- **NppStatus nppiMean\_StdDev\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp64f** \*pMean, **Npp64f** \*pStdDev)

*8-bit unsigned mean standard deviation.*

## NormDiff

Norm of pixel differences between two images.

- `NppStatus nppiNormDiff_L1_8u_C1R` (const `Npp8u` \*pSrc1, int nSrcStep1, const `Npp8u` \*pSrc2, int nSrcStep2, `NppiSize` oSizeROI, `Npp64f` \*pRetVal)  
*8-bit unsigned L1 norm of pixel differences.*
- `NppStatus nppiNormDiff_L2_8u_C1R` (const `Npp8u` \*pSrc1, int nSrcStep1, const `Npp8u` \*pSrc2, int nSrcStep2, `NppiSize` oSizeROI, `Npp64f` \*pRetVal)  
*8-bit unsigned L2 norm of pixel differences.*
- `NppStatus nppiNormDiff_Inf_8u_C1R` (const `Npp8u` \*pSrc1, int nSrcStep1, const `Npp8u` \*pSrc2, int nSrcStep2, `NppiSize` oSizeROI, `Npp64f` \*pRetVal)  
*8-bit unsigned Infinity Norm of pixel differences.*

## 1D Linear Filter

1D mask Linear Convolution Filter, with rescaling, for 8 bit images.

- `NppStatus nppiFilterColumn_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, const `Npp32s` \*pKernel, `Npp32s` nMaskSize, `Npp32s` nAnchor, `Npp32s` nDivisor)  
*8-bit unsigned 1D (column) image convolution.*
- `NppStatus nppiFilterColumn_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, const `Npp32s` \*pKernel, `Npp32s` nMaskSize, `Npp32s` nAnchor, `Npp32s` nDivisor)  
*4 channel 8-bit unsigned 1D (column) image convolution.*
- `NppStatus nppiFilterRow_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, const `Npp32s` \*pKernel, `Npp32s` nMaskSize, `Npp32s` nAnchor, `Npp32s` nDivisor)  
*8-bit unsigned 1D (row) image convolution.*
- `NppStatus nppiFilterRow_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, const `Npp32s` \*pKernel, `Npp32s` nMaskSize, `Npp32s` nAnchor, `Npp32s` nDivisor)  
*4 channel 8-bit unsigned 1D (row) image convolution.*

## 1D Window Sum

1D mask Window Sum for 8 bit images.

- `NppStatus nppiSumWindowColumn_8u32f_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp32f` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, `Npp32s` nMaskSize, `Npp32s` nAnchor)  
*8-bit unsigned 1D (column) sum to 32f.*



- `NppStatus npapiSumWindowRow_8u32f_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp32f` \*pDst, `Npp32s` nDstStep, `NppiSize` oROI, `Npp32s` nMaskSize, `Npp32s` nAnchor)

*8-bit unsigned 1D (row) sum to 32f.*

## 2D Morphology Filter

Image dilate and erod operations.

- `NppStatus npapiDilate_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*8-bit unsigned image dilation.*

- `NppStatus npapiDilate_8u_C4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*4 channel 8-bit unsigned image dilation.*

- `NppStatus npapiErode_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*8-bit unsigned image erosion.*

- `NppStatus npapiErode_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, const `Npp8u` \*pMask, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*4 channel 8-bit unsigned image erosion.*

## Convolution (2D Masks)

General purpose 2D convolution filters.

- `NppStatus npapiFilter_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, const `Npp32s` \*pKernel, `NppiSize` oKernelSize, `NppiPoint` oAnchor, `Npp32s` nDivisor)

*8-bit unsigned convolution filter.*

- `NppStatus npapiFilter_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, const `Npp32s` \*pKernel, `NppiSize` oKernelSize, `NppiPoint` oAnchor, `Npp32s` nDivisor)

*4 channel 8-bit unsigned convolution filter.*

## 2D Linear Fixed Filters

2D linear fixed filters for 8 bit images.

- `NppStatus npapiFilterBox_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*8-bit unsigned box filter.*

- `NppStatus nppiFilterBox_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*4 channel 8-bit unsigned box filter.*

## Image Rank Filters

Min, Median, and Max image filters.

- `NppStatus nppiFilterMax_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*8-bit unsigned maximum filter.*

- `NppStatus nppiFilterMax_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*4 channel 8-bit unsigned maximum filter.*

- `NppStatus nppiFilterMin_8u_C1R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*8-bit unsigned minimum filter.*

- `NppStatus nppiFilterMin_8u_C4R` (const `Npp8u` \*pSrc, `Npp32s` nSrcStep, `Npp8u` \*pDst, `Npp32s` nDstStep, `NppiSize` oSizeROI, `NppiSize` oMaskSize, `NppiPoint` oAnchor)

*4 channel 8-bit unsigned minimum filter.*

## Image Linear Transforms

Linear image transforms, like Fourier and DCT transformations.

- `NppStatus nppiMagnitude_32fc32f_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point complex to 32-bit floating point magnitude.*

- `NppStatus nppiMagnitudeSqr_32fc32f_C1R` (const `Npp32fc` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI)

*32-bit floating point complex to 32-bit floating point squared magnitude.*

## Histogram

- `NppStatus nppiEvenLevelsHost_32s` (`Npp32s` \*hpLevels, int nLevels, `Npp32s` nLowerLevel, `Npp32s` nUpperLevel)

*Compute levels with even distribution.*

- `NppStatus nppiHistogramEvenGetBufferSize_8u_C1R` (`NppiSize` oSizeROI, int nLevels, int \*hpBufferSize)

*Scratch-buffer size for `nppiHistogramEven_8u_C1R`.*

- **NppStatus nppiHistogramEven\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist, int nLevels, **Npp32s** nLowerLevel, **Npp32s** nUpperLevel, **Npp8u** \*pBuffer)  
*8-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_8u\_C4R** (**NppiSize** oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_8u\_C4R.*
- **NppStatus nppiHistogramEven\_8u\_C4R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[4], int nLevels[4], **Npp32s** nLowerLevel[4], **Npp32s** nUpperLevel[4], **Npp8u** \*pBuffer)  
*4 channel 8-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_8u\_AC4R** (**NppiSize** oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_8u\_AC4R.*
- **NppStatus nppiHistogramEven\_8u\_AC4R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[3], int nLevels[3], **Npp32s** nLowerLevel[3], **Npp32s** nUpperLevel[3], **Npp8u** \*pBuffer)  
*4 channel (alpha as the last channel) 8-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_16u\_C1R** (**NppiSize** oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16u\_C1R.*
- **NppStatus nppiHistogramEven\_16u\_C1R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist, int nLevels, **Npp32s** nLowerLevel, **Npp32s** nUpperLevel, **Npp8u** \*pBuffer)  
*16-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_16u\_C4R** (**NppiSize** oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16u\_C4R.*
- **NppStatus nppiHistogramEven\_16u\_C4R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[4], int nLevels[4], **Npp32s** nLowerLevel[4], **Npp32s** nUpperLevel[4], **Npp8u** \*pBuffer)  
*4 channel 16-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_16u\_AC4R** (**NppiSize** oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16u\_AC4R.*
- **NppStatus nppiHistogramEven\_16u\_AC4R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[3], int nLevels[3], **Npp32s** nLowerLevel[3], **Npp32s** nUpperLevel[3], **Npp8u** \*pBuffer)  
*4 channel (alpha as the last channel) 16-bit unsigned histogram with evenly distributed bins.*
- **NppStatus nppiHistogramEvenGetBufferSize\_16s\_C1R** (**NppiSize** oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16s\_C1R.*

- [NppStatus nppiHistogramEven\\_16s\\_C1R](#) (const [Npp16s](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist, int nLevels, [Npp32s](#) nLowerLevel, [Npp32s](#) nUpperLevel, [Npp8u](#) \*pBuffer)  
*16-bit signed histogram with evenly distributed bins.*
- [NppStatus nppiHistogramEvenGetBufferSize\\_16s\\_C4R](#) ([NppiSize](#) oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16s\_C4R.*
- [NppStatus nppiHistogramEven\\_16s\\_C4R](#) (const [Npp16s](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[4], int nLevels[4], [Npp32s](#) nLowerLevel[4], [Npp32s](#) nUpperLevel[4], [Npp8u](#) \*pBuffer)  
*4 channel 16-bit signed histogram with evenly distributed bins.*
- [NppStatus nppiHistogramEvenGetBufferSize\\_16s\\_AC4R](#) ([NppiSize](#) oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramEven\_16s\_AC4R.*
- [NppStatus nppiHistogramEven\\_16s\\_AC4R](#) (const [Npp16s](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[3], int nLevels[3], [Npp32s](#) nLowerLevel[3], [Npp32s](#) nUpperLevel[3], [Npp8u](#) \*pBuffer)  
*4 channel (alpha as the last channel) 16-bit signed histogram with evenly distributed bins.*
- [NppStatus nppiHistogramRangeGetBufferSize\\_8u\\_C1R](#) ([NppiSize](#) oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_8u\_C1R.*
- [NppStatus nppiHistogramRange\\_8u\\_C1R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist, const [Npp32s](#) \*pLevels, int nLevels, [Npp8u](#) \*pBuffer)  
*8-bit unsigned histogram with bins determined by pLevels array.*
- [NppStatus nppiHistogramRangeGetBufferSize\\_8u\\_C4R](#) ([NppiSize](#) oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_8u\_C4R.*
- [NppStatus nppiHistogramRange\\_8u\\_C4R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[4], const [Npp32s](#) \*pLevels[4], int nLevels[4], [Npp8u](#) \*pBuffer)  
*4 channel 8-bit unsigned histogram with bins determined by pLevels.*
- [NppStatus nppiHistogramRangeGetBufferSize\\_8u\\_AC4R](#) ([NppiSize](#) oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_8u\_AC4R.*
- [NppStatus nppiHistogramRange\\_8u\\_AC4R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[3], const [Npp32s](#) \*pLevels[3], int nLevels[3], [Npp8u](#) \*pBuffer)  
*4 channel (alpha as a last channel) 8-bit unsigned histogram with bins determined by pLevels.*
- [NppStatus nppiHistogramRangeGetBufferSize\\_16u\\_C1R](#) ([NppiSize](#) oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16u\_C1R.*

- **NppStatus nppiHistogramRange\_16u\_C1R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist, const **Npp32s** \*pLevels, int nLevels, **Npp8u** \*pBuffer)  
*16-bit unsigned histogram with bins determined by pLevels array.*
- **NppStatus nppiHistogramRangeGetBufferSize\_16u\_C4R** (**NppiSize** oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16u\_C4R.*
- **NppStatus nppiHistogramRange\_16u\_C4R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[4], const **Npp32s** \*pLevels[4], int nLevels[4], **Npp8u** \*pBuffer)  
*4 channel 16-bit unsigned histogram with bins determined by pLevels.*
- **NppStatus nppiHistogramRangeGetBufferSize\_16u\_AC4R** (**NppiSize** oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16u\_AC4R.*
- **NppStatus nppiHistogramRange\_16u\_AC4R** (const **Npp16u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[3], const **Npp32s** \*pLevels[3], int nLevels[3], **Npp8u** \*pBuffer)  
*4 channel (alpha as a last channel) 16-bit unsigned histogram with bins determined by pLevels.*
- **NppStatus nppiHistogramRangeGetBufferSize\_16s\_C1R** (**NppiSize** oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16s\_C1R.*
- **NppStatus nppiHistogramRange\_16s\_C1R** (const **Npp16s** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist, const **Npp32s** \*pLevels, int nLevels, **Npp8u** \*pBuffer)  
*16-bit signed histogram with bins determined by pLevels array.*
- **NppStatus nppiHistogramRangeGetBufferSize\_16s\_C4R** (**NppiSize** oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16s\_C4R.*
- **NppStatus nppiHistogramRange\_16s\_C4R** (const **Npp16s** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[4], const **Npp32s** \*pLevels[4], int nLevels[4], **Npp8u** \*pBuffer)  
*4 channel 16-bit signed histogram with bins determined by pLevels.*
- **NppStatus nppiHistogramRangeGetBufferSize\_16s\_AC4R** (**NppiSize** oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_16s\_AC4R.*
- **NppStatus nppiHistogramRange\_16s\_AC4R** (const **Npp16s** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist[3], const **Npp32s** \*pLevels[3], int nLevels[3], **Npp8u** \*pBuffer)  
*4 channel (alpha as a last channel) 16-bit signed histogram with bins determined by pLevels.*
- **NppStatus nppiHistogramRangeGetBufferSize\_32f\_C1R** (**NppiSize** oSizeROI, int nLevels, int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_32f\_C1R.*
- **NppStatus nppiHistogramRange\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pHist, const **Npp32f** \*pLevels, int nLevels, **Npp8u** \*pBuffer)  
*32-bit float histogram with bins determined by pLevels array.*

- [NppStatus nppiHistogramRangeGetBufferSize\\_32f\\_C4R](#) ([NppiSize](#) oSizeROI, int nLevels[4], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_32f\_C4R.*
- [NppStatus nppiHistogramRange\\_32f\\_C4R](#) (const [Npp32f](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[4], const [Npp32f](#) \*pLevels[4], int nLevels[4], [Npp8u](#) \*pBuffer)  
*4 channel 32-bit float histogram with bins determined by pLevels.*
- [NppStatus nppiHistogramRangeGetBufferSize\\_32f\\_AC4R](#) ([NppiSize](#) oSizeROI, int nLevels[3], int \*hpBufferSize)  
*Scratch-buffer size for nppiHistogramRange\_32f\_AC4R.*
- [NppStatus nppiHistogramRange\\_32f\\_AC4R](#) (const [Npp32f](#) \*pSrc, int nSrcStep, [NppiSize](#) oSizeROI, [Npp32s](#) \*pHist[3], const [Npp32f](#) \*pLevels[3], int nLevels[3], [Npp8u](#) \*pBuffer)  
*4 channel (alpha as a last channel) 32-bit float histogram with bins determined by pLevels.*

## JPEG DCT, Quantization and Level Shift Functions

Jpeg standard defines a flow of level shift, DCT and quantization for forward JPEG transform and inverse level shift, IDCT and de-quantization for inverse JPEG transform.

This group has the functions for both forward and inverse functions.

- [NppStatus nppiQuantFwdRawTableInit\\_JPEG\\_8u](#) ([Npp8u](#) \*pQuantRawTable, int nQualityFactor)  
*Converts regular quantization tables with the quality factor.*
- [NppStatus nppiQuantFwdTableInit\\_JPEG\\_8u16u](#) (const [Npp8u](#) \*pQuantRawTable, [Npp16u](#) \*pQuantFwdRawTable)  
*Converts raw quantization table to a forward quantization table.*
- [NppStatus nppiQuantInvTableInit\\_JPEG\\_8u16u](#) (const [Npp8u](#) \*pQuantRawTable, [Npp16u](#) \*pQuantFwdRawTable)  
*Converts raw quantization table to an inverse quantization table.*
- [NppStatus nppiSetDefaultQuantTable](#) ([Npp8u](#) \*pQuantRawTable, int tableIndex)  
*Fills out the quantization table with either luminance and chrominance tables for JPEG.*
- [NppStatus nppiDCTQuantInv8x8LS\\_JPEG\\_16s8u\\_C1R](#) ([Npp16s](#) \*pSrc, int nSrcStep, [Npp8u](#) \*pDst, int nDstStep, const [Npp16u](#) \*pQuantInvTable, [NppiSize](#) oSizeROI)  
*Inverse DCT, de-quantization and level shift part of the JPEG decoding.*
- [NppStatus nppiDCTQuantFwd8x8LS\\_JPEG\\_8u16s\\_C1R](#) ([Npp8u](#) \*pSrc, int nSrcStep, [Npp16s](#) \*pDst, int nDstStep, const [Npp16u](#) \*pQuantFwdTable, [NppiSize](#) oSizeROI)  
*Forward DCT, quantization and level shift part of the JPEG encoding.*

## Sum

Sum of 8 bit images.

- **NppStatus** **nppiReductionGetBufferHostSize\_8u\_C1R** (const **NppiSize** &oSizeROI, int \*hpBufferSize)  
*Scratch-buffer size for nppiSum\_8u\_C1R.*
- **NppStatus** **nppiReductionGetBufferHostSize\_8u\_C4R** (const **NppiSize** &oSizeROI, int \*hpBufferSize)  
*Scratch-buffer size for nppiSum\_8u\_C4R.*
- **NppStatus** **nppiSum\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pDeviceBuffer, **Npp64f** \*pSum)  
*8-bit unsigned image sum.*
- **NppStatus** **nppiSum\_8u\_C4R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp32s** \*pDeviceBuffer, **Npp64f** aSum[4])  
*4 channel 8-bit unsigned image sum.*

## MinMax

Minimum and maximum of 8-bit images.

- **NppStatus** **nppiMinMax\_8u\_C1R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp8u** \*pMin, **Npp8u** \*pMax)  
*8-bit unsigned pixel minimum and maximum.*
- **NppStatus** **nppiMinMax\_8u\_C4R** (const **Npp8u** \*pSrc, int nSrcStep, **NppiSize** oSizeROI, **Npp8u** cuMin[4], **Npp8u** cuMax[4])  
*4 channel 8-bit unsigned pixel minimum and maximum.*

## Resize

Resizes 8 bit images.

Handles C1 and C4 images.

- **NppStatus** **nppiResize\_8u\_C1R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcROI, **Npp8u** \*pDst, int nDstStep, **NppiSize** dstROISize, double xFactor, double yFactor, int interpolation)  
*8-bit unsigned image resize.*
- **NppStatus** **nppiResize\_8u\_C4R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcROI, **Npp8u** \*pDst, int nDstStep, **NppiSize** dstROISize, double xFactor, double yFactor, int interpolation)  
*4 channel 8-bit unsigned image resize.*

## Rotate

Rotates an image around the origin (0,0) and then shifts it.

- `NppStatus nppiRotate_8u_C1R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcROI, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstROI, double angle, double xShift, double yShift, int interpolation)  
*8-bit unsigned image rotate.*
- `NppStatus nppiRotate_8u_C4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcROI, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstROI, double angle, double xShift, double yShift, int interpolation)  
*4 channel 8-bit unsigned image rotate.*

## Mirror

Mirrors images horizontally, vertically and diagonally.

- `NppStatus nppiMirror_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oROI, `NppiAxis` flip)  
*8-bit unsigned image mirror.*
- `NppStatus nppiMirror_8u_C4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oROI, `NppiAxis` flip)  
*4 channel 8-bit unsigned image mirror.*

## RGBToYCbCr

RGB to YCbCr color conversion.

- `NppStatus nppiRGBToYCbCr_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned packed RGB to packed YCbCr color conversion.*
- `NppStatus nppiRGBToYCbCr422_8u_C3C2R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned RGB to 2 channel chroma packed YCbCr422 color conversion.*
- `NppStatus nppiRGBToYCbCr420_8u_C3P3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*\*pDst, int nDstStep[3], `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned packed RGB to planar YCbCr420 color conversion.*
- `NppStatus nppiRGBToYCbCr_8u_P3R` (const `Npp8u` \*const \*pSrc, int nSrcStep, `Npp8u` \*\*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel planar 8-bit unsigned RGB to YCbCr color conversion.*
- `NppStatus nppiRGBToYCbCr_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned RGB to YCbCr color conversion, ignoring Alpha.*



## YCbCrToRGB

YCbCr to RGB color conversion.

- `NppStatus nppiYCbCrToRGB_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned packed YCbCr to RGB color conversion.*
- `NppStatus nppiYCbCrToRGB_8u_P3R` (const `Npp8u` \*const \*pSrc, int nSrcStep, `Npp8u` \*\*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr to RGB color conversion.*
- `NppStatus nppiYCbCrToRGB_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*4 channel 8-bit unsigned packed YCbCr to RGB color conversion, not affecting Alpha.*
- `NppStatus nppiYCbCr422ToRGB_8u_C2C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*2 channel 8-bit unsigned YCbCr422 to 3 channel packed RGB color conversion.*
- `NppStatus nppiYCbCr420ToRGB_8u_P3C3R` (const `Npp8u` \*const \*pSrc, int nSrcStep[3], `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr420 to packed RGB color conversion.*

## Sample Pattern Conversion.

- `NppStatus nppiYCbCr422ToYCbCr420_8u_P3R` (const `Npp8u` \*const \*pSrc, int nSrcStep[3], `Npp8u` \*\*pDst, int nDstStep[3], `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr:422 to YCbCr:420 resampling.*
- `NppStatus nppiYCbCr422ToYCbCr411_8u_P3R` (const `Npp8u` \*const \*pSrc, int nSrcStep[3], `Npp8u` \*\*pDst, int nDstStep[3], `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr:422 to YCbCr:411 resampling.*
- `NppStatus nppiYCbCr420ToYCbCr422_8u_P3R` (const `Npp8u` \*const \*pSrc, int nSrcStep[3], `Npp8u` \*\*pDst, int nDstStep[3], `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr:420 to YCbCr:422 resampling.*
- `NppStatus nppiYCbCr420ToYCbCr411_8u_P3P2R` (const `Npp8u` \*const \*pSrc, int aSrcStep[3], `Npp8u` \*pDstY, int nDstYStep, `Npp8u` \*pDstCbCr, int nDstCbCrStep, `NppiSize` oSizeROI)  
*3 channel 8-bit unsigned planar YCbCr:420 to YCbCr:411 resampling.*

## Color Processing

Color manipulation functions.

- `NppStatus nppiColorTwist32f_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` twist[3][4])

*3 channel 8-bit unsigned color twist.*

- `NppStatus nppiColorTwist32f_8u_P3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*\*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` twist[3][4])

*3 channel planar 8-bit unsigned color twist.*

- `NppStatus nppiColorTwist32f_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` twist[3][4])

*4 channel 8-bit unsigned color twist, not affecting Alpha.*

- `NppStatus nppiLUT_Linear_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32s` \*pValues, const `Npp32s` \*pLevels, int nLevels)

*8-bit unsigned look-up-table color conversion.*

- `NppStatus nppiLUT_Linear_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32s` \*pValues[3], const `Npp32s` \*pLevels[3], int nLevels[3])

*3 channel 8-bit unsigned look-up-table color conversion.*

- `NppStatus nppiLUT_Linear_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32s` \*pValues[4], const `Npp32s` \*pLevels[4], int nLevels[4])

*4 channel 8-bit unsigned look-up-table color conversion, not affecting Alpha.*

## Canny Edge Detection

- `NppStatus nppiCannyGetBufferSize` (`NppiSize` oSizeROI, int \*hpBufferSize)

*Calculate the size of a temporary buffer needed for the Canny function.*

- `NppStatus nppiCanny_32f8u_C1R` (const `Npp32f` \*pSrcDx, int nSrcDxStep, const `Npp32f` \*pSrcDy, int nSrcDyStep, `Npp8u` \*pDstEdges, int nDstEdgeStep, `NppiSize` oSizeROI, `Npp32f` nLowThreshold, `Npp32f` nHighThreshold, `Npp8u` \*pBuffer)

*Canny edge detection.*

## Affine warping, affine transform calculation

Affine warping of an image is the transform of image pixel positions, defined by the following formulas:

$$X_{new} = C_{00} * x + C_{01} * y + C_{02} \quad Y_{new} = C_{10} * x + C_{11} * y + C_{12} \quad C = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \end{bmatrix}$$

That is, any pixel with coordinates  $(X_{new}, Y_{new})$  in the transformed image is sourced from coordinates  $(x, y)$  in the original image.

The mapping  $C$  is completely specified by 6 values  $C_{ij}, i = \overline{0,1}, j = \overline{0,2}$ . The transform maps parallel lines to parallel lines and preserves ratios of distances of points to lines. Implementation specific properties are discussed in each function's documentation.

- `NppStatus nppiGetAffineTransform` (`NppiRect` srcRoi, const double quad[4][2], double coeffs[2][3])

*Calculates affine transform coefficients given source rectangular ROI and its destination quadrangle projection.*

- **NppStatus nppiGetAffineQuad** (**NppiRect** srcRoi, double quad[4][2], const double coeffs[2][3])  
*Calculates affine transform projection of given source rectangular ROI.*
- **NppStatus nppiGetAffineBound** (**NppiRect** srcRoi, double bound[2][2], const double coeffs[2][3])  
*Calculates bounding box of the affine transform projection of the given source rectangular ROI.*
- **NppStatus nppiWarpAffine\_8u\_C1R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, single channel).*
- **NppStatus nppiWarpAffine\_8u\_C3R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, three channels).*
- **NppStatus nppiWarpAffine\_8u\_C4R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, four channels).*
- **NppStatus nppiWarpAffine\_8u\_AC4R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, four channels RGBA).*
- **NppStatus nppiWarpAffine\_8u\_P3R** (const **Npp8u** \*pSrc[3], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst[3], int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, three planes).*
- **NppStatus nppiWarpAffine\_8u\_P4R** (const **Npp8u** \*pSrc[4], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst[4], int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Affine transform of an image (8bit unsigned integer, four planes).*
- **NppStatus nppiWarpAffineBack\_8u\_C1R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Inverse affine transform of an image (8bit unsigned integer, single channel).*
- **NppStatus nppiWarpAffineBack\_8u\_C3R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Inverse affine transform of an image (8bit unsigned integer, three channels).*
- **NppStatus nppiWarpAffineBack\_8u\_C4R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)  
*Inverse affine transform of an image (8bit unsigned integer, four channels).*
- **NppStatus nppiWarpAffineBack\_8u\_AC4R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (8bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpAffineBack_8u_P3R` (const `Npp8u *pSrc[3]`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, `Npp8u *pDst[3]`, int `nDstStep`, `NppiRect` `dstRoi`, const double `coeffs[2][3]`, int interpolation)

*Inverse affine transform of an image (8bit unsigned integer, three planes).*

- `NppStatus nppiWarpAffineBack_8u_P4R` (const `Npp8u *pSrc[4]`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, `Npp8u *pDst[4]`, int `nDstStep`, `NppiRect` `dstRoi`, const double `coeffs[2][3]`, int interpolation)

*Inverse affine transform of an image (8bit unsigned integer, four planes).*

- `NppStatus nppiWarpAffineQuad_8u_C1R` (const `Npp8u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, single channel).*

- `NppStatus nppiWarpAffineQuad_8u_C3R` (const `Npp8u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, three channels).*

- `NppStatus nppiWarpAffineQuad_8u_C4R` (const `Npp8u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, four channels).*

- `NppStatus nppiWarpAffineQuad_8u_AC4R` (const `Npp8u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpAffineQuad_8u_P3R` (const `Npp8u *pSrc[3]`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst[3]`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, three planes).*

- `NppStatus nppiWarpAffineQuad_8u_P4R` (const `Npp8u *pSrc[4]`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, const double `srcQuad[4][2]`, `Npp8u *pDst[4]`, int `nDstStep`, `NppiRect` `dstRoi`, const double `dstQuad[4][2]`, int interpolation)

*Affine transform of an image (8bit unsigned integer, four planes).*

- `NppStatus nppiWarpAffine_16u_C1R` (const `Npp16u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, `Npp16u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `coeffs[2][3]`, int interpolation)

*Affine transform of an image (16bit unsigned integer, single channel).*

- `NppStatus nppiWarpAffine_16u_C3R` (const `Npp16u *pSrc`, `NppiSize` `srcSize`, int `nSrcStep`, `NppiRect` `srcRoi`, `Npp16u *pDst`, int `nDstStep`, `NppiRect` `dstRoi`, const double `coeffs[2][3]`, int interpolation)

*Affine transform of an image (16bit unsigned integer, three channels).*

- `NppStatus nppiWarpAffine_16u_C4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (16bit unsigned integer, four channels).*

- `NppStatus nppiWarpAffine_16u_AC4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (16bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpAffine_16u_P3R` (const `Npp16u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (16bit unsigned integer, three planes).*

- `NppStatus nppiWarpAffine_16u_P4R` (const `Npp16u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (16bit unsigned integer, four planes).*

- `NppStatus nppiWarpAffineBack_16u_C1R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, single channel).*

- `NppStatus nppiWarpAffineBack_16u_C3R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, three channels).*

- `NppStatus nppiWarpAffineBack_16u_C4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, four channels).*

- `NppStatus nppiWarpAffineBack_16u_AC4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpAffineBack_16u_P3R` (const `Npp16u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, three planes).*

- `NppStatus nppiWarpAffineBack_16u_P4R` (const `Npp16u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (16bit unsigned integer, four planes).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_C1R** (const **Npp16u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, single channel).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_C3R** (const **Npp16u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, three channels).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_C4R** (const **Npp16u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, four channels).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_AC4R** (const **Npp16u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, four channels RGBA).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_P3R** (const **Npp16u** \*pSrc[3], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst[3], int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, three planes).*

- **NppStatus** **nppiWarpAffineQuad\_16u\_P4R** (const **Npp16u** \*pSrc[4], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp16u** \*pDst[4], int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (16bit unsigned integer, four planes).*

- **NppStatus** **nppiWarpAffine\_32f\_C1R** (const **Npp32f** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp32f** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, single channel).*

- **NppStatus** **nppiWarpAffine\_32f\_C3R** (const **Npp32f** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp32f** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, three channels).*

- **NppStatus** **nppiWarpAffine\_32f\_C4R** (const **Npp32f** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp32f** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, four channels).*

- **NppStatus** **nppiWarpAffine\_32f\_AC4R** (const **Npp32f** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp32f** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, four channels RGBA).*

- `NppStatus nppiWarpAffine_32f_P3R` (const `Npp32f *pSrc[3]`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst[3]`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpAffine_32f_P4R` (const `Npp32f *pSrc[4]`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst[4]`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpAffineBack_32f_C1R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, single channel).*

- `NppStatus nppiWarpAffineBack_32f_C3R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, three channels).*

- `NppStatus nppiWarpAffineBack_32f_C4R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, four channels).*

- `NppStatus nppiWarpAffineBack_32f_AC4R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, four channels RGBA).*

- `NppStatus nppiWarpAffineBack_32f_P3R` (const `Npp32f *pSrc[3]`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst[3]`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpAffineBack_32f_P4R` (const `Npp32f *pSrc[4]`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f *pDst[4]`, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpAffineQuad_32f_C1R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, single channel).*

- `NppStatus nppiWarpAffineQuad_32f_C3R` (const `Npp32f *pSrc`, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f *pDst`, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, three channels).*



- `NppStatus nppiWarpAffineQuad_32f_C4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, four channels).*

- `NppStatus nppiWarpAffineQuad_32f_AC4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, four channels RGBA).*

- `NppStatus nppiWarpAffineQuad_32f_P3R` (const `Npp32f` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpAffineQuad_32f_P4R` (const `Npp32f` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpAffine_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpAffine_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, three channels).*

- `NppStatus nppiWarpAffine_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpAffine_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, four channels RGBA).*

- `NppStatus nppiWarpAffine_32s_P3R` (const `Npp32s` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, three planes).*

- `NppStatus nppiWarpAffine_32s_P4R` (const `Npp32s` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Affine transform of an image (32bit signed integer, four planes).*



- `NppStatus nppiWarpAffineBack_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpAffineBack_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, three channels).*

- `NppStatus nppiWarpAffineBack_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpAffineBack_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, four channels RGBA).*

- `NppStatus nppiWarpAffineBack_32s_P3R` (const `Npp32s` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, three planes).*

- `NppStatus nppiWarpAffineBack_32s_P4R` (const `Npp32s` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[2][3], int interpolation)

*Inverse affine transform of an image (32bit signed integer, four planes).*

- `NppStatus nppiWarpAffineQuad_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpAffineQuad_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, three channels).*

- `NppStatus nppiWarpAffineQuad_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpAffineQuad_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, four channels RGBA).*

- **NppStatus** **nppiWarpAffineQuad\_32s\_P3R** (const **Npp32s** \*pSrc[3], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp32s** \*pDst[3], int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, three planes).*

- **NppStatus** **nppiWarpAffineQuad\_32s\_P4R** (const **Npp32s** \*pSrc[4], **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, const double srcQuad[4][2], **Npp32s** \*pDst[4], int nDstStep, **NppiRect** dstRoi, const double dstQuad[4][2], int interpolation)

*Affine transform of an image (32bit signed integer, four planes).*

## Perspective warping, perspective transform calculation

Perspective warping of an image is the transform of image pixel positions, defined by the following formula:

$$X_{new} = \frac{C_{00} * x + C_{01} * y + C_{02}}{C_{20} * x + C_{21} * y + C_{22}} \quad Y_{new} = \frac{C_{10} * x + C_{11} * y + C_{12}}{C_{20} * x + C_{21} * y + C_{22}} \quad C = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix}$$

That is, any pixel of the transformed image with coordinates  $(X_{new}, Y_{new})$  has a preimage with coordinates  $(x, y)$ .

The mapping  $C$  is fully defined by 8 values  $C_{ij}, (i, j) = \overline{0, 2}$ , except of  $C_{22}$ , which is a normalizer. The transform has a property of mapping any convex quadrangle to a convex quadrangle, which is used in a group of functions **nppiWarpPerspectiveQuad**. The NPPI implementation of perspective transform has some issues which are discussed in each function's documentation.

- **NppStatus** **nppiGetPerspectiveTransform** (**NppiRect** srcRoi, const double quad[4][2], double coeffs[3][3])

*Calculates perspective transform coefficients given source rectangular ROI and its destination quadrangle projection.*

- **NppStatus** **nppiGetPerspectiveQuad** (**NppiRect** srcRoi, double quad[4][2], const double coeffs[3][3])

*Calculates perspective transform projection of given source rectangular ROI.*

- **NppStatus** **nppiGetPerspectiveBound** (**NppiRect** srcRoi, double bound[2][2], const double coeffs[3][3])

*Calculates bounding box of the perspective transform projection of the given source rectangular ROI.*

- **NppStatus** **nppiWarpPerspective\_8u\_C1R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, single channel).*

- **NppStatus** **nppiWarpPerspective\_8u\_C3R** (const **Npp8u** \*pSrc, **NppiSize** srcSize, int nSrcStep, **NppiRect** srcRoi, **Npp8u** \*pDst, int nDstStep, **NppiRect** dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspective_8u_C4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspective_8u_AC4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspective_8u_P3R` (const `Npp8u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspective_8u_P4R` (const `Npp8u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspectiveBack_8u_C1R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, single channel).*

- `NppStatus nppiWarpPerspectiveBack_8u_C3R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspectiveBack_8u_C4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspectiveBack_8u_AC4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveBack_8u_P3R` (const `Npp8u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspectiveBack_8u_P4R` (const `Npp8u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp8u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (8bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspectiveQuad_8u_C1R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, single channel).*

- `NppStatus nppiWarpPerspectiveQuad_8u_C3R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspectiveQuad_8u_C4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspectiveQuad_8u_AC4R` (const `Npp8u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveQuad_8u_P3R` (const `Npp8u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspectiveQuad_8u_P4R` (const `Npp8u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp8u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (8bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspective_16u_C1R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, single channel).*

- `NppStatus nppiWarpPerspective_16u_C3R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspective_16u_C4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspective_16u_AC4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspective_16u_P3R` (const `Npp16u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspective_16u_P4R` (const `Npp16u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspectiveBack_16u_C1R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, single channel).*

- `NppStatus nppiWarpPerspectiveBack_16u_C3R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspectiveBack_16u_C4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspectiveBack_16u_AC4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveBack_16u_P3R` (const `Npp16u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspectiveBack_16u_P4R` (const `Npp16u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp16u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (16bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspectiveQuad_16u_C1R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, single channel).*

- `NppStatus nppiWarpPerspectiveQuad_16u_C3R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, three channels).*

- `NppStatus nppiWarpPerspectiveQuad_16u_C4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four channels).*

- `NppStatus nppiWarpPerspectiveQuad_16u_AC4R` (const `Npp16u` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveQuad_16u_P3R` (const `Npp16u` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, three planes).*

- `NppStatus nppiWarpPerspectiveQuad_16u_P4R` (const `Npp16u` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp16u` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (16bit unsigned integer, four planes).*

- `NppStatus nppiWarpPerspective_32f_C1R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, single channel).*

- `NppStatus nppiWarpPerspective_32f_C3R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, three channels).*

- `NppStatus nppiWarpPerspective_32f_C4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, four channels).*

- `NppStatus nppiWarpPerspective_32f_AC4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, four channels RGBA).*

- `NppStatus nppiWarpPerspective_32f_P3R` (const `Npp32f` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpPerspective_32f_P4R` (const `Npp32f` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpPerspectiveBack_32f_C1R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, single channel).*

- `NppStatus nppiWarpPerspectiveBack_32f_C3R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, three channels).*

- `NppStatus nppiWarpPerspectiveBack_32f_C4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, four channels).*

- `NppStatus nppiWarpPerspectiveBack_32f_AC4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveBack_32f_P3R` (const `Npp32f` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpPerspectiveBack_32f_P4R` (const `Npp32f` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32f` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpPerspectiveQuad_32f_C1R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, single channel).*

- `NppStatus nppiWarpPerspectiveQuad_32f_C3R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, three channels).*

- `NppStatus nppiWarpPerspectiveQuad_32f_C4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, four channels).*

- `NppStatus nppiWarpPerspectiveQuad_32f_AC4R` (const `Npp32f` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, four channels RGBA).*



- `NppStatus nppiWarpPerspectiveQuad_32f_P3R` (const `Npp32f` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, three planes).*

- `NppStatus nppiWarpPerspectiveQuad_32f_P4R` (const `Npp32f` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32f` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit float, four planes).*

- `NppStatus nppiWarpPerspective_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpPerspective_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, three channels).*

- `NppStatus nppiWarpPerspective_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpPerspective_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, four channels RGBA).*

- `NppStatus nppiWarpPerspective_32s_P3R` (const `Npp32s` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, three planes).*

- `NppStatus nppiWarpPerspective_32s_P4R` (const `Npp32s` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Perspective transform of an image (32bit signed integer, four planes).*

- `NppStatus nppiWarpPerspectiveBack_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpPerspectiveBack_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, three channels).*



- `NppStatus nppiWarpPerspectiveBack_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpPerspectiveBack_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveBack_32s_P3R` (const `Npp32s` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, three planes).*

- `NppStatus nppiWarpPerspectiveBack_32s_P4R` (const `Npp32s` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, `Npp32s` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double coeffs[3][3], int interpolation)

*Inverse perspective transform of an image (32bit signed integer, four planes).*

- `NppStatus nppiWarpPerspectiveQuad_32s_C1R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, single channel).*

- `NppStatus nppiWarpPerspectiveQuad_32s_C3R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, three channels).*

- `NppStatus nppiWarpPerspectiveQuad_32s_C4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, four channels).*

- `NppStatus nppiWarpPerspectiveQuad_32s_AC4R` (const `Npp32s` \*pSrc, `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst, int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, four channels RGBA).*

- `NppStatus nppiWarpPerspectiveQuad_32s_P3R` (const `Npp32s` \*pSrc[3], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst[3], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, three planes).*

- `NppStatus nppiWarpPerspectiveQuad_32s_P4R` (const `Npp32s` \*pSrc[4], `NppiSize` srcSize, int nSrcStep, `NppiRect` srcRoi, const double srcQuad[4][2], `Npp32s` \*pDst[4], int nDstStep, `NppiRect` dstRoi, const double dstQuad[4][2], int interpolation)

*Perspective transform of an image (32bit signed integer, four planes).*

## Image Labeling Techniques

- [NppStatus nppiGraphcutGetSize](#) ([NppiSize](#) size, int \*pBufSize)  
*Calculates the size of the temporary buffer for graph-cut labeling.*
- [NppStatus nppiGraphcut\\_32s8u](#) ([Npp32s](#) \*pTerminals, [Npp32s](#) \*pLeftTransposed, [Npp32s](#) \*pRightTransposed, [Npp32s](#) \*pTop, [Npp32s](#) \*pBottom, int nStep, int nTransposedStep, [NppiSize](#) size, [Npp8u](#) \*pLabel, int nLabelStep, [Npp8u](#) \*pBuffer)  
*Graphcut of a flow network (32bit signed integer edge capacities).*

### 7.4.1 Function Documentation

#### 7.4.1.1 [NppStatus nppiAbsDiff\\_32f\\_C1R](#) (const [Npp32f](#) \*pSrc1, int nSrc1Step, const [Npp32f](#) \*pSrc2, int nSrc2Step, [Npp32f](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI)

32-bit floating point absolute difference.

Compute abs(sourcePixel1 - sourcePixel2).

##### Parameters:

[pSrc1](#) [Source-Image Pointer](#).  
[nSrc1Step](#) [Source-Image Line Step](#).  
[pSrc2](#) [Source-Image Pointer](#).  
[nSrc2Step](#) [Source-Image Line Step](#).  
[pDst](#) [Destination-Image Pointer](#).  
[nDstStep](#) [Destination-Image Line Step](#).  
[oSizeROI](#) [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.2 [NppStatus nppiAbsDiff\\_32s\\_C1R](#) (const [Npp32s](#) \*pSrc1, int nSrc1Step, const [Npp32s](#) \*pSrc2, int nSrc2Step, [Npp32s](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI)

32-bit absolute difference.

Compute abs(sourcePixel1 - sourcePixel2).

##### Parameters:

[pSrc1](#) [Source-Image Pointer](#).  
[nSrc1Step](#) [Source-Image Line Step](#).  
[pSrc2](#) [Source-Image Pointer](#).  
[nSrc2Step](#) [Source-Image Line Step](#).  
[pDst](#) [Destination-Image Pointer](#).  
[nDstStep](#) [Destination-Image Line Step](#).  
[oSizeROI](#) [Region-of-Interest \(ROI\)](#).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.3 NppStatus npDiffAbs\_8u\_AC4R (const Npp8u \* pSrc1, int nSrc1Step, const Npp8u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 8-bit unsigned absolute difference, not affecting Alpha.

Compute abs(sourcePixel1 - sourcePixel2).

**Parameters:**

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.4 NppStatus npDiffAbs\_8u\_C1R (const Npp8u \* pSrc1, int nSrc1Step, const Npp8u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

8-bit unsigned absolute difference.

Compute abs(sourcePixel1 - sourcePixel2).

**Parameters:**

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.5 NppStatus nppiAbsDiff\_8u\_C4R (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 8-bit unsigned absolute difference.

Compute  $\text{abs}(\text{sourcePixel1} - \text{sourcePixel2})$ .

##### Parameters:

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.6 NppStatus nppiAbsDiffC\_32f\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, Npp32f *nValue*)

32-bit floating point image absolute difference from constant.

##### Parameters:

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nValue* Constant.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.7 NppStatus nppiAdd\_32f\_C1R (const Npp32f \* *pSrc1*, int *nSrc1Step*, const Npp32f \* *pSrc2*, int *nSrc2Step*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit floating point image add.

Add the pixel values of corresponding pixels in the ROI and write them to the output image.

##### Parameters:

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.8 NppStatus npplAdd\_32s\_C1R (const Npp32s \* *pSrc1*, int *nSrc1Step*, const Npp32s \* *pSrc2*, int *nSrc2Step*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit image add.

Add the pixel values of corresponding pixels in the ROI and write them to the output image.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.9 NppStatus npplAdd\_8u\_AC4RSfs (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

4 channel 8-bit unsigned image add, not affecting Alpha.

Add the pixel values of corresponding pixels in the ROI and write them to the output image.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{-(nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.10 NppStatus nppiAdd\_8u\_C1RSfs (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

8-bit unsigned image add.

Add the pixel values of corresponding pixels in the ROI and write them to the output image.

##### Parameters:

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.11 NppStatus nppiAdd\_8u\_C4RSfs (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

4 channel 8-bit unsigned image add.

Add the pixel values of corresponding pixels in the ROI and write them to the output image.

##### Parameters:

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.12 NppStatus nppiAddC\_32f\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f *nValue*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit floating point image add constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.13 NppStatus nppiAddC\_32fc\_C1R (const Npp32fc \* *pSrc*, int *nSrcStep*, Npp32fc *nValue*, Npp32fc \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit complex floating point image add constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.14 NppStatus nppiCanny\_32f8u\_C1R (const Npp32f \* *pSrcDx*, int *nSrcDxStep*, const Npp32f \* *pSrcDy*, int *nSrcDyStep*, Npp8u \* *pDstEdges*, int *nDstEdgeStep*, NppiSize *oSizeROI*, Npp32f *nLowThreshold*, Npp32f *nHighThreshold*, Npp8u \* *pBuffer*)

Canny edge detection.

Find edges using the Canny algorithm. This function requires a temporary working device buffer whose size should be computed by a prior call to nppiCannyGetBufferSize.

**Parameters:**

*pSrcDx* Pointer to the source image ROI x-derivative.  
*nSrcDxStep* Distance in bytes between starts of consecutive lines in the source image *pSrcDx*.  
*pSrcDy* Pointer to the source image ROI y-derivative.  
*nSrcDyStep* Distance in bytes between starts of consecutive lines in the source image *pSrcDy*.  
*pDstEdges* Pointer to the output array of the detected edges.  
*nDstEdgeStep* Distance in bytes between starts of consecutive lines in the output image.  
*oSizeROI* Width and height of the regions of interest.

***nLowThreshold*** Lower threshold for edge detection.

***nHighThreshold*** Upper threshold for edge detection.

***pBuffer*** Pointer to the pre-allocated temporary buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_BAD\\_ARG\\_ERROR](#) if lowThresh is negative or highThresh is less than lowThresh.
- [NPP\\_NOT\\_EVEN\\_STEP\\_ERROR](#) if any step size is not divisible by 4.
- [NPP\\_WARNING](#) if the edge detection did not converge within the maximum number of iterations

#### 7.4.1.15 NppStatus nppiCannyGetBufferSize (NppiSize oSizeROI, int \* hpBufferSize)

Calculate the size of a temporary buffer needed for the Canny function.

**Parameters:**

***oSizeROI*** Size of the image ROI in pixels

***hpBufferSize*** Host-pointer receiving the size required temporary buffer.

**Returns:**

Error codes:

- [NPP\\_NULL\\_POINTER\\_ERROR](#) indicates an error condition if pBufSize point is NULL
- [NPP\\_SIZE\\_ERROR](#) if oSizeROI has a field with zero or negative value

#### 7.4.1.16 NppStatus nppiColorTwist32f\_8u\_AC4R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f twist[3][4])

4 channel 8-bit unsigned color twist, not affecting Alpha.

An input color twist matrix with floating-point pixel values is applied with in ROI. Alpha channel is the last channel and is not processed.

**Parameters:**

***pSrc*** Source-Image Pointer.

***nSrcStep*** Source-Image Line Step.

***pDst*** Destination-Image Pointer.

***nDstStep*** Destination-Image Line Step.

***oSizeROI*** Region-of-Interest (ROI).

***twist*** The color twist matrix with floating-point pixel values.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)



#### 7.4.1.17 **NppStatus nppiColorTwist32f\_8u\_C3R** (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp32f *twist*[3][4])

3 channel 8-bit unsigned color twist.

An input color twist matrix with floating-point pixel values is applied within ROI.

##### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*twist* The color twist matrix with floating-point pixel values.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.18 **NppStatus nppiColorTwist32f\_8u\_P3R** (const Npp8u \* *const* \* *pSrc*, int *nSrcStep*, Npp8u \*\* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp32f *twist*[3][4])

3 channel planar 8-bit unsigned color twist.

An input color twist matrix with floating-point pixel values is applied within ROI.

##### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*twist* The color twist matrix with floating-point pixel values.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.19 **NppStatus nppiCompare\_32f\_C1R** (const Npp32f \* *pSrc1*, int *nSrc1Step*, const Npp32f \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, NppCmpOp *eComparisonOperation*)

32-bit floating point image compare.

Compare *pSrc1*'s pixels with corresponding pixels in *pSrc2*.

##### Parameters:

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*eComparisonOperation* Specifies the comparison operation to be used in the pixel comparison.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.20** `NppStatus nppiCompare_8u_AC4R (const Npp8u * pSrc1, int nSrc1Step, const Npp8u * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned image compare, not affecting Alpha.

Compare pSrc1's pixels with corresponding pixels in pSrc2.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*eComparisonOperation* Specifies the comparison operation to be used in the pixel comparison.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.21** `NppStatus nppiCompare_8u_C4R (const Npp8u * pSrc1, int nSrc1Step, const Npp8u * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned image compare.

Compare pSrc1's pixels with corresponding pixels in pSrc2.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*eComparisonOperation* Specifies the comparison operation to be used in the pixel comparison.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.22 NppStatus nppiConvert\_16s32f\_C1R (const Npp16s \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI)**

16-bit singedto 32-bit floating point conversion.

For detailed documentation see [nppiConverte\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.23 NppStatus nppiConvert\_16s32s\_C1R (const Npp16s \* pSrc, int nSrcStep, Npp32s \* pDst, int nDstStep, NppiSize oSizeROI)**

16-bit to 32-bit conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.24 **NppStatus nppiConvert\_16s8u\_AC4R** (const Npp16s \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 16-bit signed to 8-bit unsigned conversion, not affecting Alpha.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

##### Parameters:

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.25 **NppStatus nppiConvert\_16s8u\_C1R** (const Npp16s \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 16-bit signed to 8-bit unsigned conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

##### Parameters:

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.26 **NppStatus nppiConvert\_16s8u\_C4R** (const Npp16s \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 16-bit signed to 8-bit unsigned conversion, not affecting Alpha.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

##### Parameters:

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.27 NppStatus nppiConvert\_16u32f\_C1R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

16-bit unsigned to 32-bit floating point conversion.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.28 NppStatus nppiConvert\_16u32s\_C1R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

16-bit unsigned to 32-bit signed conversion.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.29 NppStatus nppiConvert\_16u8u\_AC4R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 16-bit unsigned to 8-bit unsigned conversion, not affecting Alpha.

For detailed documentation see `nppiConvert_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.30 NppStatus nppiConvert\_16u8u\_C1R (const Npp16u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

16-bit unsigned to 8-bit unsigned conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.31 NppStatus nppiConvert\_16u8u\_C4R (const Npp16u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 16-bit unsigned to 8-bit unsigned conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.32 NppStatus nppiConvert\_32f16s\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, NppRoundMode *eRoundMode*)**

32-bit floating point to 16-bit conversion.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*eRoundMode* Flag specifying how fractional float values are rounded to integer values.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.33 NppStatus nppiConvert\_32f16u\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, NppRoundMode *eRoundMode*)**

32-bit floating point to 16-bit unsigned conversion.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*eRoundMode* Flag specifying how fractional float values are rounded to integer values.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.34 NppStatus nppiConvert\_32f8u\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, NppRoundMode *eRoundMode*)**

32-bit floating point to 8-bit unsigned conversion.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*eRoundMode* Flag specifying how fractional float values are rounded to integer values.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.35 NppStatus nppiConvert\_8u16s\_AC4R (const Npp8u \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 8-bit unsigned to 16-bit signed conversion, not affecting Alpha.

For detailed documentation see `nppiConverte_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.36 NppStatus nppiConvert\_8u16s\_C1R (const Npp8u \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI)**

8-bit unsigned to 16-bit signed conversion.

For detailed documentation see `nppiConvert_8u16u_C1R()`.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes



**7.4.1.37 NppStatus nppiConvert\_8u16s\_C4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 8-bit unsigned to 16-bit signed conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.38 NppStatus nppiConvert\_8u16u\_AC4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 8-bit unsigned to 16-bit unsigned conversion, not affecting Alpha.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.39 NppStatus nppiConvert\_8u16u\_C1R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

8-bit unsigned to 16-bit unsigned conversion.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.40 NppStatus nppiConvert\_8u16u\_C4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 8-bit unsigned to 16-bit unsigned conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.41 NppStatus nppiConvert\_8u32f\_C1R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

8-bit unsigned to 32-bit floating point conversion.

For detailed documentation see [nppiConvert\\_8u16u\\_C1R\(\)](#).

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.42 NppStatus nppiCopy\_16s\_AC4R (const Npp16s \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 16-bit image copy, not affecting Alpha.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.43 NppStatus nppiCopy\_16s\_C1R (const Npp16s \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

16-bit image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.44 NppStatus nppiCopy\_16s\_C4R (const Npp16s \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 16-bit image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.45 NppStatus nppiCopy\_16u\_AC4R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 16-bit unsigned image copy, not affecting Alpha channel.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.46 NppStatus nppiCopy\_16u\_C1R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

16-bit unsigned image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.47 NppStatus nppiCopy\_16u\_C4R (const Npp16u \* *pSrc*, int *nSrcStep*, Npp16u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 16-bit unsigned image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.48 NppStatus nppiCopy\_32f\_AC4R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 32-bit floating point image copy, not affecting Alpha.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.49 NppStatus nppiCopy\_32f\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

32-bit floating point image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.50 NppStatus nppiCopy\_32f\_C4R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 32-bit floating point image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.51 NppStatus nppiCopy\_32s\_AC4R (const Npp32s \* *pSrc*, int *nSrcStep*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 32-bit image copy, not affecting Alpha.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.52 NppStatus nppiCopy\_32s\_C1R (const Npp32s \* *pSrc*, int *nSrcStep*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

32-bit image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.53 NppStatus nppiCopy\_32s\_C4R (const Npp32s \* *pSrc*, int *nSrcStep*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 32-bit image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.54 NppStatus nppiCopy\_8u\_AC4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 8-bit unsigned image copy, not affecting Alpha channel.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.55 NppStatus nppiCopy\_8u\_C1R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

8-bit unsigned image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.56 NppStatus nppiCopy\_8u\_C4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)**

4 channel 8-bit unsigned image copy.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.57 NppStatus nppiCopyConstBorder\_32s\_C1R (const Npp32s \* *pSrc*, int *nSrcStep*, NppiSize *oSrcSizeROI*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oDstSizeROI*, int *nTopBorderHeight*, int *nLeftBorderWidth*, Npp32s *nValue*)**

32-bit image copy with constant border color.

See [nppiCopyConstBorder\\_8u\\_C1R\(\)](#) for detailed documentation.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*oSrcSizeROI* Size of the source region-of-interest.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.

*oDstSizeROI* Size of the destination region-of-interest.

*nTopBorderHeight* Height of top border.

*nLeftBorderWidth* Width of left border.

*nValue* Border luminance value.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.58** `NppStatus nppiCopyConstBorder_8u_AC4R (const Npp8u * pSrc, int nSrcStep, NppiSize oSrcSizeROI, Npp8u * pDst, int nDstStep, NppiSize oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, const Npp8u aValue[3])`

4 channel 8-bit unsigned image copy with constant border color.

See [nppiCopyConstBorder\\_8u\\_C1R\(\)](#) for detailed documentation.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSrcSizeROI* Size of the source region-of-interest.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oDstSizeROI* Size of the destination region-of-interest.

*nTopBorderHeight* Height of top border.

*nLeftBorderWidth* Width of left border.

*aValue* Vector of the RGB values of the border pixels. Because this method does not affect the destination image's alpha channel, only three components of the border color are needed.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.59** `NppStatus nppiCopyConstBorder_8u_C1R (const Npp8u * pSrc, int nSrcStep, NppiSize oSrcSizeROI, Npp8u * pDst, int nDstStep, NppiSize oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, Npp8u nValue)`

8-bit unsigned image copy width constant border color.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSrcSizeROI* Size of the source region of pixels.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oDstSizeROI* Size (width, height) of the destination region, i.e. the region that gets filled with data from the source image (inner part) and constant border color (outer part).



***nTopBorderHeight*** Height (in pixels) of the top border. The height of the border at the bottom of the destination ROI is implicitly defined by the size of the source ROI:  $nBottomBorderHeight = oDstSizeROI.height - nTopBorderHeight - oSrcSizeROI.height$ .

***nLeftBorderWidth*** Width (in pixels) of the left border. The width of the border at the right side of the destination ROI is implicitly defined by the size of the source ROI:  $nRightBorderWidth = oDstSizeROI.width - nLeftBorderWidth - oSrcSizeROI.width$ .

***nValue*** The pixel value to be set for border pixels.

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.60 NppStatus nppiCopyConstBorder\_8u\_C4R (const Npp8u \*pSrc, int nSrcStep, NppiSize oSrcSizeROI, Npp8u \*pDst, int nDstStep, NppiSize oDstSizeROI, int nTopBorderHeight, int nLeftBorderWidth, const Npp8u aValue[4])

4channel 8-bit unsigned image copy with constant border color.

See [nppiCopyConstBorder\\_8u\\_C1R\(\)](#) for detailed documentation.

#### Parameters:

***pSrc*** [Source-Image Pointer](#).

***nSrcStep*** [Source-Image Line Step](#).

***oSrcSizeROI*** Size of the source region-of-interest.

***pDst*** [Destination-Image Pointer](#).

***nDstStep*** [Destination-Image Line Step](#).

***oDstSizeROI*** Size of the destination region-of-interest.

***nTopBorderHeight*** Height of top border.

***nLeftBorderWidth*** Width of left border.

***aValue*** Vector of the RGBA values of the border pixels to be set.

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.61 NppStatus nppiDCTQuantFwd8x8LS\_JPEG\_8u16s\_C1R (Npp8u \*pSrc, int nSrcStep, Npp16s \*pDst, int nDstStep, const Npp16u \*pQuantFwdTable, NppiSize oSizeROI)

Forward DCT, quantization and level shift part of the JPEG encoding.

Input is expected in 8x8 macro blocks and output is expected to be in 64x1 macro blocks.

#### Parameters:

***pSrc*** [Source-Image Pointer](#).

***nSrcStep*** [Source-Image Line Step](#).

***pDst*** [Destination-Image Pointer](#).

***nDstStep*** [Destination-Image Line Step](#).

*pQuantFwdTable* Forward quantization tables for JPEG encoding.

*oSizeROI* Region-of-Interest (ROI).

#### Returns:

Error codes:

- **NPP\_SIZE\_ERROR** For negative input height/width or not a multiple of 8 width/height.
- **NPP\_STEP\_ERROR** If input image width is not multiple of 8 or does not match ROI.
- **NPP\_NULL\_POINTER\_ERROR** If the destination pointer is NULL.

#### 7.4.1.62 NppStatus nppiDCTQuantInv8x8LS\_JPEG\_16s8u\_C1R (Npp16s \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, const Npp16u \* pQuantInvTable, NppiSize oSizeROI)

Inverse DCT, de-quantization and level shift part of the JPEG decoding.

Input is expected in 64x1 macro blocks and output is expected to be in 8x8 macro blocks.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*pQuantInvTable* Inverse quantization tables for JPEG decoding.

*oSizeROI* Region-of-Interest (ROI).

#### Returns:

Error codes:

- **NPP\_SIZE\_ERROR** For negative input height/width or not a multiple of 8 width/height.
- **NPP\_STEP\_ERROR** If input image width is not multiple of 8 or does not match ROI.
- **NPP\_NULL\_POINTER\_ERROR** If the destination pointer is NULL.

#### 7.4.1.63 NppStatus nppiDilate\_8u\_C1R (const Npp8u \* pSrc, Npp32s nSrcStep, Npp8u \* pDst, Npp32s nDstStep, NppiSize oSizeROI, const Npp8u \* pMask, NppiSize oMaskSize, NppiPoint oAnchor)

8-bit unsigned image dilation.

Dilation computes the output pixel as the maximum pixel value of the pixels under the mask. Pixels whose corresponding mask values are zero do not participate in the maximum search.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the start address of the mask array

*oMaskSize* Width and Height mask array.

*oAnchor* X and Y offsets of the mask origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.64** `NppStatus nppiDilate_8u_C4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, NppiSize oMaskSize, NppiPoint oAnchor)`

4 channel 8-bit unsigned image dilation.

Dilation computes the output pixel as the maximum pixel value of the pixels under the mask. Pixels whose corresponding mask values are zero do not participate in the maximum search.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the start address of the mask array

*oMaskSize* Width and Height mask array.

*oAnchor* X and Y offsets of the mask origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.65** `NppStatus nppiDiv_32f_C1R (const Npp32f * pSrc1, int nSrc1Step, const Npp32f * pSrc2, int nSrc2Step, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point image division.

Divide pixels in pSrc2 by pSrc1's pixels.

**Parameters:**

*pSrc1* [Source-Image Pointer](#).

*nSrc1Step* [Source-Image Line Step](#).

*pSrc2* [Source-Image Pointer](#).

*nSrc2Step* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.66 **NppStatus nppiDiv\_32s\_C1R** (const Npp32s \* *pSrc1*, int *nSrc1Step*, const Npp32s \* *pSrc2*, int *nSrc2Step*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit image division.

Divide pixels in *pSrc2* by *pSrc1*'s pixels.

##### Parameters:

*pSrc1* [Source-Image Pointer](#).  
*nSrc1Step* [Source-Image Line Step](#).  
*pSrc2* [Source-Image Pointer](#).  
*nSrc2Step* [Source-Image Line Step](#).  
*pDst* [Destination-Image Pointer](#).  
*nDstStep* [Destination-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.67 **NppStatus nppiDiv\_8u\_AC4RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

4 channel 8-bit unsigned image division, not affecting Alpha.

Divide pixels in *pSrc2* by *pSrc1*'s pixels.

##### Parameters:

*pSrc1* [Source-Image Pointer](#).  
*nSrc1Step* [Source-Image Line Step](#).  
*pSrc2* [Source-Image Pointer](#).  
*nSrc2Step* [Source-Image Line Step](#).  
*pDst* [Destination-Image Pointer](#).  
*nDstStep* [Destination-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.68 **NppStatus nppiDiv\_8u\_C1RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

8-bit unsigned image division.

Divided pixels in *pSrc2* by *pSrc1*'s pixels.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.69** `NppStatus nppiDiv_8u_C4RSfs (const Npp8u * pSrc1, int nSrc1Step, const Npp8u * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, int nScaleFactor)`

4 channel 8-bit unsigned image division.

Divide pixels in pSrc2 by pSrc1's pixels.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.70** `NppStatus nppiDivC_32f_C1R (const Npp32f * pSrc, int nSrcStep, Npp32f nValue, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point image divide by constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*nValue* Constant.  
*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.71** `NppStatus nppiDivC_32fc_C1R (const Npp32fc * pSrc, int nSrcStep, Npp32fc nValue, Npp32fc * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit complex floating point image divide by constant.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.72** `NppStatus nppiErode_8u_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oSizeROI, const Npp8u * pMask, NppiSize oMaskSize, NppiPoint oAnchor)`

8-bit unsigned image erosion.

Erosion computes the output pixel as the minimum pixel value of the pixels under the mask. Pixels whose corresponding mask values are zero do not participate in the maximum search.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pMask* Pointer to the start address of the mask array

*oMaskSize* Width and Height mask array.

*oAnchor* X and Y offsets of the mask origin frame of reference w.r.t the source pixel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.73 **NppStatus nppiErode\_8u\_C4R** (const Npp8u \* *pSrc*, Npp32s *nSrcStep*, Npp8u \* *pDst*, Npp32s *nDstStep*, NppiSize *oSizeROI*, const Npp8u \* *pMask*, NppiSize *oMaskSize*, NppiPoint *oAnchor*)

4 channel 8-bit unsigned image erosion.

Erosion computes the output pixel as the minimum pixel value of the pixels under the mask. Pixels whose corresponding mask values are zero do not participate in the maximum search.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the start address of the mask array

*oMaskSize* Width and Height mask array.

*oAnchor* X and Y offsets of the mask origin frame of reference w.r.t the source pixel.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.74 **NppStatus nppiEvenLevelsHost\_32s** (Npp32s \* *hpLevels*, int *nLevels*, Npp32s *nLowerLevel*, Npp32s *nUpperLevel*)

Compute levels with even distribution.

##### Parameters:

*hpLevels* A host pointer to array which receives the levels being computed. The array needs to be of size *nLevels*.

*nLevels* The number of levels being computed. *nLevels* must be at least 2, otherwise an NPP\_HISTO\_NUMBER\_OF\_LEVELS\_ERROR error is returned.

*nLowerLevel* Lower boundary value of the lowest level.

*nUpperLevel* Upper boundary value of the greatest level.

##### Returns:

Error code.

#### 7.4.1.75 **NppStatus nppiExp\_32f\_C1R** (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit floating point exponentiation.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.76** `NppStatus nppiFilter_8u_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oSizeROI, const Npp32s * pKernel, NppiSize oKernelSize, NppiPoint oAnchor, Npp32s nDivisor)`

8-bit unsigned convolution filter.

Pixels under the mask are multiplied by the respective weights in the mask and the results are summed. Before writing the result pixel the sum is scaled back via division by nDivisor.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.  
*oKernelSize* Width and Height of the rectangular kernel.  
*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.  
*nDivisor* The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.77** `NppStatus nppiFilter_8u_C4R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oSizeROI, const Npp32s * pKernel, NppiSize oKernelSize, NppiPoint oAnchor, Npp32s nDivisor)`

4 channel 8-bit unsigned convolution filter.

Pixels under the mask are multiplied by the respective weights in the mask and the results are summed. Before writing the result pixel the sum is scaled back via division by nDivisor.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.



*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.

*oKernelSize* Width and Height of the rectangular kernel.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

*nDivisor* The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.78** `NppStatus nppiFilterBox_8u_C1R (const Npp8u *pSrc, Npp32s nSrcStep, Npp8u *pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)`

8-bit unsigned box filter.

Computes the average pixel values of the pixels under a rectangular mask.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*oMaskSize* Width and Height of the neighborhood region for the local Avg operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.79** `NppStatus nppiFilterBox_8u_C4R (const Npp8u *pSrc, Npp32s nSrcStep, Npp8u *pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)`

4 channel 8-bit unsigned box filter.

Computes the average pixel values of the pixels under a rectangular mask.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*oMaskSize* Width and Height of the neighborhood region for the local Avg operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.80** `NppStatus nppiFilterColumn_8u_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oROI, const Npp32s * pKernel, Npp32s nMaskSize, Npp32s nAnchor, Npp32s nDivisor)`

8-bit unsigned 1D (column) image convolution.

Apply convolution filter with user specified 1D column of weights. Result pixel is equal to the sum of the products between the kernel coefficients (pKernel array) and corresponding neighboring column pixel values in the source image defined by nKernelDim and nAnchorY, divided by nDivisor.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oROI* [Region-of-Interest \(ROI\)](#).

*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.

*nMaskSize* Length of the linear kernel array.

*nAnchor* Y offset of the kernel origin frame of reference w.r.t the source pixel.

*nDivisor* The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.81** `NppStatus nppiFilterColumn_8u_C4R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oROI, const Npp32s * pKernel, Npp32s nMaskSize, Npp32s nAnchor, Npp32s nDivisor)`

4 channel 8-bit unsigned 1D (column) image convolution.

Apply convolution filter with user specified 1D column of weights. Result pixel is equal to the sum of the products between the kernel coefficients (pKernel array) and corresponding neighboring column pixel values in the source image defined by nKernelDim and nAnchorY, divided by nDivisor.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oROI* Region-of-Interest (ROI).

*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.

*nMaskSize* Length of the linear kernel array.

*nAnchor* Y offset of the kernel origin frame of reference w.r.t the source pixel.

*nDivisor* The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.82 NppStatus nppiFilterMax\_8u\_C1R (const Npp8u \* pSrc, Npp32s nSrcStep, Npp8u \* pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)

8-bit unsigned maximum filter.

Result pixel value is the maximum of pixel values under the rectangular mask region.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*oMaskSize* Width and Height of the neighborhood region for the local Max operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.83 NppStatus nppiFilterMax\_8u\_C4R (const Npp8u \* pSrc, Npp32s nSrcStep, Npp8u \* pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)

4 channel 8-bit unsigned maximum filter.

Result pixel value is the maximum of pixel values under the rectangular mask region.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*oMaskSize* Width and Height of the neighborhood region for the local Max operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.84** `NppStatus nppiFilterMin_8u_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)`

8-bit unsigned minimum filter.

Result pixel value is the minimum of pixel values under the rectangular mask region.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*oMaskSize* Width and Height of the neighborhood region for the local Max operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.85** `NppStatus nppiFilterMin_8u_C4R (const Npp8u * pSrc, Npp32s nSrcStep, Npp8u * pDst, Npp32s nDstStep, NppiSize oSizeROI, NppiSize oMaskSize, NppiPoint oAnchor)`

4 channel 8-bit unsigned minimum filter.

Result pixel value is the minimum of pixel values under the rectangular mask region.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*oMaskSize* Width and Height of the neighborhood region for the local Max operation.

*oAnchor* X and Y offsets of the kernel origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.86 NppStatus nppiFilterRow\_8u\_C1R (const Npp8u \* *pSrc*, Npp32s *nSrcStep*, Npp8u \* *pDst*, Npp32s *nDstStep*, NppiSize *oROI*, const Npp32s \* *pKernel*, Npp32s *nMaskSize*, Npp32s *nAnchor*, Npp32s *nDivisor*)**

8-bit unsigned 1D (row) image convolution.

Apply general linear Row convolution filter, with rescaling, in a 1D mask region around each source pixel for 1-channel 8 bit/pixel images. Result pixel is equal to the sum of the products between the kernel coefficients (*pKernel* array) and corresponding neighboring row pixel values in the source image defined by *iKernelDim* and *iAnchorX*, divided by *iDivisor*.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oROI* [Region-of-Interest \(ROI\)](#).

*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.

*nMaskSize* Length of the linear kernel array.

*nAnchor* X offset of the kernel origin frame of reference w.r.t the source pixel.

*nDivisor* The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.87 NppStatus nppiFilterRow\_8u\_C4R (const Npp8u \* *pSrc*, Npp32s *nSrcStep*, Npp8u \* *pDst*, Npp32s *nDstStep*, NppiSize *oROI*, const Npp32s \* *pKernel*, Npp32s *nMaskSize*, Npp32s *nAnchor*, Npp32s *nDivisor*)**

4 channel 8-bit unsigned 1D (row) image convolution.

Apply general linear Row convolution filter, with rescaling, in a 1D mask region around each source pixel for 1-channel 8 bit/pixel images. Result pixel is equal to the sum of the products between the kernel coefficients (*pKernel* array) and corresponding neighboring row pixel values in the source image defined by *iKernelDim* and *iAnchorX*, divided by *iDivisor*.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oROI* [Region-of-Interest \(ROI\)](#).

*pKernel* Pointer to the start address of the kernel coefficient array. Coefficients are expected to be stored in reverse order.

*nMaskSize* Length of the linear kernel array.

***nAnchor*** X offset of the kernel origin frame of reference w.r.t the source pixel.

***nDivisor*** The factor by which the convolved summation from the Filter operation should be divided. If equal to the sum of coefficients, this will keep the maximum result value within full scale.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.88 void nppiFree (void \* *pData*)**

Free method for any 2D allocated memory.

This method should be used to free memory allocated with any of the nppiMalloc\_<modifier> methods.

**Parameters:**

***pData*** A pointer to memory allocated using nppiMalloc\_<modifier>.

**7.4.1.89 NppStatus nppiGetAffineBound (NppiRect *srcRoi*, double *bound*[2][2], const double *coeffs*[2][3])**

Calculates bounding box of the affine transform projection of the given source rectangular ROI.

**Parameters:**

***srcRoi*** Source ROI

***bound*** Bounding box of the transformed source ROI

***coeffs*** Affine transform coefficients

**Returns:**

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

**7.4.1.90 NppStatus nppiGetAffineQuad (NppiRect *srcRoi*, double *quad*[4][2], const double *coeffs*[2][3])**

Calculates affine transform projection of given source rectangular ROI.

**Parameters:**

***srcRoi*** Source ROI

***quad*** Destination quadrangle

***coeffs*** Affine transform coefficients

**Returns:**

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

**7.4.1.91 NppStatus nppiGetAffineTransform (NppiRect *srcRoi*, const double *quad*[4][2], double *coeffs*[2][3])**

Calculates affine transform coefficients given source rectangular ROI and its destination quadrangle projection.

**Parameters:**

*srcRoi* Source ROI

*quad* Destination quadrangle

*coeffs* Affine transform coefficients

**Returns:**

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_AFFINE\\_QUAD\\_INCORRECT\\_WARNING](#) Indicates a warning when *quad* does not conform to the transform properties. Fourth vertex is ignored, internally computed coordinates are used instead

**7.4.1.92 NppStatus nppiGetPerspectiveBound (NppiRect *srcRoi*, double *bound*[2][2], const double *coeffs*[3][3])**

Calculates bounding box of the perspective transform projection of the given source rectangular ROI.

**Parameters:**

*srcRoi* Source ROI

*bound* Bounding box of the transformed source ROI

*coeffs* Perspective transform coefficients

**Returns:**

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

#### 7.4.1.93 NppStatus nppiGetPerspectiveQuad (NppiRect *srcRoi*, double *quad*[4][2], const double *coeffs*[3][3])

Calculates perspective transform projection of given source rectangular ROI.

##### Parameters:

*srcRoi* Source ROI  
*quad* Destination quadrangle  
*coeffs* Perspective transform coefficients

##### Returns:

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

#### 7.4.1.94 NppStatus nppiGetPerspectiveTransform (NppiRect *srcRoi*, const double *quad*[4][2], double *coeffs*[3][3])

Calculates perspective transform coefficients given source rectangular ROI and its destination quadrangle projection.

##### Parameters:

*srcRoi* Source ROI  
*quad* Destination quadrangle  
*coeffs* Perspective transform coefficients

##### Returns:

Error codes:

- [NPP\\_SIZE\\_ERROR](#) Indicates an error condition if any image dimension has zero or negative value
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

#### 7.4.1.95 NppStatus nppiGraphcut\_32s8u (Npp32s \* *pTerminals*, Npp32s \* *pLeftTransposed*, Npp32s \* *pRightTransposed*, Npp32s \* *pTop*, Npp32s \* *pBottom*, int *nStep*, int *nTransposedStep*, NppiSize *size*, Npp8u \* *pLabel*, int *nLabelStep*, Npp8u \* *pBuffer*)

Graphcut of a flow network (32bit signed integer edge capacities).

The function computes the minimal cut (graphcut) of a 2D regular 4-connected graph. The inputs are the capacities of the horizontal (in transposed form), vertical and terminal (source and sink) edges. The capacities to source and sink are stored as capacity differences in the terminals array ( *terminals*(x) =



source(x) - sink(x) ). The implementation assumes that the edge capacities for boundary edges that would connect to nodes outside the specified domain are set to 0 (for example left(0,\*) == 0). If this is not fulfilled the computed labeling may be wrong! The computed binary labeling is encoded as unsigned 8bit values (0 / 255).

**See also:**

[nppiGraphcutGetSize](#)

**Parameters:**

*pTerminals* Pointer to differences of terminal edge capacities (terminal(x) = source(x) - sink(x))

*pLeftTransposed* Pointer to transposed left edge capacities (left(0,\*) must be 0)

*pRightTransposed* Pointer to transposed right edge capacities (right(width-1,\*) must be 0)

*pTop* Pointer to top edge capacities (top(\*,0) must be 0)

*pBottom* Pointer to bottom edge capacities (bottom(\*,height-1) must be 0)

*nStep* Step in bytes between any pair of sequential rows of edge capacities

*nTransposedStep* Step in bytes between any pair of sequential rows of transposed edge capacities

*size* Graph size

*pLabel* Pointer to destination label image

*nLabelStep* Step in bytes between any pair of sequential rows of label image

*pBuffer* Pointer to the temporary buffer

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.96 NppStatus nppiGraphcutGetSize (NppiSize size, int \* pBufSize)

Calculates the size of the temporary buffer for graph-cut labeling.

**See also:**

[nppiGraphcut\\_32s8u](#)

**Parameters:**

*size* Graph size

*pBufSize* Pointer to variable that returns the size of the temporary buffer.

**Returns:**

NPP\_SUCCESS Indicates no error. Any other value indicates an error or a warning

NPP\_SIZE\_ERROR Indicates an error condition if any image dimension has zero or negative value

NPP\_NULL\_POINTER\_ERROR Indicates an error condition if pBufSize pointer is NULL

**7.4.1.97** `NppStatus nppiHistogramEven_16s_AC4R (const Npp16s * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[3], int nLevels[3], Npp32s nLowerLevel[3], Npp32s nUpperLevel[3], Npp8u * pBuffer)`

4 channel (alpha as the last channel) 16-bit signed histogram with evenly distributed bins.

Alpha channel is ignored during histogram computation.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving computed histograms per color channel. Array pointed by pHist[i] be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*nLowerLevel* Array containing lower-level of lowest bin per color channel.

*nUpperLevel* Array containing upper-level of highest bin per color channel.

*pBuffer* Pointer to appropriately sized (nppiHistogramEvenGetBufferSize\_16s\_AC4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.98** `NppStatus nppiHistogramEven_16s_C1R (const Npp16s * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, int nLevels, Npp32s nLowerLevel, Npp32s nUpperLevel, Npp8u * pBuffer)`

16-bit signed histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Pointer to array that receives the computed histogram. The array must be of size nLevels-1.

*nLevels* Number of levels.

*nLowerLevel* Lower boundary of lowest level bin.

*nUpperLevel* Upper boundary of highest level bin.

*pBuffer* Pointer to appropriately sized (nppiHistogramEvenGetBufferSize\_16s\_C1R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.99** `NppStatus nppiHistogramEven_16s_C4R (const Npp16s * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], int nLevels[4], Npp32s nLowerLevel[4], Npp32s nUpperLevel[4], Npp8u * pBuffer)`

4 channel 16-bit signed histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving computed histograms per color channel. Array pointed by pHist[i] be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*nLowerLevel* Array containing lower-level of lowest bin per color channel.

*nUpperLevel* Array containing upper-level of highest bin per color channel.

*pBuffer* Pointer to appropriately sized (nppiHistogramEvenGetBufferSize\_16s\_C4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.100** `NppStatus nppiHistogramEven_16u_AC4R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[3], int nLevels[3], Npp32s nLowerLevel[3], Npp32s nUpperLevel[3], Npp8u * pBuffer)`

4 channel (alpha as the last channel) 16-bit unsigned histogram with evenly distributed bins.

Alpha channel is ignored during histogram computation.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving computed histograms per color channel. Array pointed by pHist[i] be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*nLowerLevel* Array containing lower-level of lowest bin per color channel.

*nUpperLevel* Array containing upper-level of highest bin per color channel.

*pBuffer* Pointer to appropriately sized (nppiHistogramEvenGetBufferSize\_16u\_AC4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.101** `NppStatus nppiHistogramEven_16u_C1R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, int nLevels, Npp32s nLowerLevel, Npp32s nUpperLevel, Npp8u * pBuffer)`

16-bit unsigned histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Pointer to array that receives the computed histogram. The array must be of size nLevels-1.

*nLevels* Number of levels.

*nLowerLevel* Lower boundary of lowest level bin.

*nUpperLevel* Upper boundary of highest level bin.

*pBuffer* Pointer to appropriately sized (`nppiHistogramEvenGetBufferSize_16u_C1R`) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.102** `NppStatus nppiHistogramEven_16u_C4R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], int nLevels[4], Npp32s nLowerLevel[4], Npp32s nUpperLevel[4], Npp8u * pBuffer)`

4 channel 16-bit unsigned histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving computed histograms per color channel. Array pointed by `pHist[i]` be of size `nLevels[i]-1`.

*nLevels* Array containing number of levels per color channel.

*nLowerLevel* Array containing lower-level of lowest bin per color channel.

*nUpperLevel* Array containing upper-level of highest bin per color channel.

*pBuffer* Pointer to appropriately sized (`nppiHistogramEvenGetBufferSize_16u_C4R`) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.103** `NppStatus nppiHistogramEven_8u_AC4R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[3], int nLevels[3], Npp32s nLowerLevel[3], Npp32s nUpperLevel[3], Npp8u * pBuffer)`

4 channel (alpha as the last channel) 8-bit unsigned histogram with evenly distributed bins.

Alpha channel is ignored during histogram computation.

**Parameters:**

*pSrc* [Source-Image Pointer](#).  
*nSrcStep* [Source-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*pHist* Array of pointers which are receiving computed histograms per color channel. Array pointed by *pHist[i]* be of size *nLevels[i]*-1.  
*nLevels* Array containing number of levels per color channel.  
*nLowerLevel* Array containing lower-level of lowest bin per color channel.  
*nUpperLevel* Array containing upper-level of highest bin per color channel.  
*pBuffer* Pointer to appropriately sized (`nppiHistogramEvenGetBufferSize_8u_AC4R`) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.104** `NppStatus nppiHistogramEven_8u_C1R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, int nLevels, Npp32s nLowerLevel, Npp32s nUpperLevel, Npp8u * pBuffer)`

8-bit unsigned histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).  
*nSrcStep* [Source-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*pHist* Pointer to array that receives the computed histogram. The array must be of size *nLevels*-1.  
*nLevels* Number of levels.  
*nLowerLevel* Lower boundary of lowest level bin.  
*nUpperLevel* Upper boundary of highest level bin.  
*pBuffer* Pointer to appropriately sized (`nppiHistogramEvenGetBufferSize_8u_C1R`) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.105** `NppStatus nppiHistogramEven_8u_C4R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], int nLevels[4], Npp32s nLowerLevel[4], Npp32s nUpperLevel[4], Npp8u * pBuffer)`

4 channel 8-bit unsigned histogram with evenly distributed bins.

**Parameters:**

*pSrc* [Source-Image Pointer](#).  
*nSrcStep* [Source-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).

***pHist*** Array of pointers which are receiving computed histograms per color channel. Array pointed by *pHist[i]* be of size *nLevels[i]-1*.

***nLevels*** Array containing number of levels per color channel.

***nLowerLevel*** Array containing lower-level of lowest bin per color channel.

***nUpperLevel*** Array containing upper-level of highest bin per color channel.

***pBuffer*** Pointer to appropriately sized (*npPiHistogramEvenGetBufferSize\_8u\_C4R*) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.106 NppStatus npPiHistogramEvenGetBufferSize\_16s\_AC4R (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)**

Scratch-buffer size for *npPiHistogramEven\_16s\_AC4R*.

**Parameters:**

***oSizeROI*** ROI size.

***nLevels*** Array containing number of levels per color channel.

***hpBufferSize*** Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.107 NppStatus npPiHistogramEvenGetBufferSize\_16s\_C1R (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)**

Scratch-buffer size for *npPiHistogramEven\_16s\_C1R*.

**Parameters:**

***oSizeROI*** [Region-of-Interest \(ROI\)](#).

***nLevels*** Number of levels in the histogram.

***hpBufferSize*** Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.108 NppStatus npPiHistogramEvenGetBufferSize\_16s\_C4R (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)**

Scratch-buffer size for *npPiHistogramEven\_16s\_C4R*.

**Parameters:**

***oSizeROI*** ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.109 NppStatus nppiHistogramEvenGetBufferSize\_16u\_AC4R (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramEven\_16u\_AC4R.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.110 NppStatus nppiHistogramEvenGetBufferSize\_16u\_C1R (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramEven\_16u\_C1R.

**Parameters:**

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nLevels* Number of levels in the histogram.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.111 NppStatus nppiHistogramEvenGetBufferSize\_16u\_C4R (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramEven\_16u\_C4R.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

#### 7.4.1.112 **NppStatus nppiHistogramEvenGetBufferSize\_8u\_AC4R** (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)

Scratch-buffer size for nppiHistogramEven\_8u\_AC4R.

##### Parameters:

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.113 **NppStatus nppiHistogramEvenGetBufferSize\_8u\_C1R** (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)

Scratch-buffer size for nppiHistogramEven\_8u\_C1R.

##### Parameters:

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nLevels* Number of levels in the histogram.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.114 **NppStatus nppiHistogramEvenGetBufferSize\_8u\_C4R** (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)

Scratch-buffer size for nppiHistogramEven\_8u\_C4R.

##### Parameters:

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.115 **NppStatus nppiHistogramRange\_16s\_AC4R** (const Npp16s \* *pSrc*, int *nSrcStep*, NppiSize *oSizeROI*, Npp32s \* *pHist*[3], const Npp32s \* *pLevels*[3], int *nLevels*[3], Npp8u \* *pBuffer*)

4 channel (alpha as a last channel) 16-bit signed histogram with bins determined by *pLevels*.

Alpha channel is ignored during the histograms computations.



**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by pHist[i] must be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by pLevel[i] must be of size nLevels[i].

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_16\_AC4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.116** `NppStatus nppiHistogramRange_16s_C1R (const Npp16s * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, const Npp32s * pLevels, int nLevels, Npp8u * pBuffer)`

16-bit signed histogram with bins determined by pLevels array.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Pointer to array that receives the computed histogram. The array must be of size nLevels-1.

*pLevels* Pointer to array containing the level sizes of the bins. The array must be of size nLevels.

*nLevels* Number of levels in histogram.

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_16\_C1R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.117** `NppStatus nppiHistogramRange_16s_C4R (const Npp16s * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], const Npp32s * pLevels[4], int nLevels[4], Npp8u * pBuffer)`

4 channel 16-bit signed histogram with bins determined by pLevels.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

***pHist*** Array of pointers which are receiving the computed histograms per color channel. Array pointed by *pHist[i]* must be of size *nLevels[i]*-1.

***nLevels*** Array containing number of levels per color channel.

***pLevels*** Array containing pointers to level-arrays per color channel. Array pointed by *pLevel[i]* must be of size *nLevels[i]*.

***pBuffer*** Pointer to appropriately sized (`npPiHistogramRangeGetBufferSize_16s_C4R`) scratch buffer.

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.118** `NppStatus npPiHistogramRange_16u_AC4R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[3], const Npp32s * pLevels[3], int nLevels[3], Npp8u * pBuffer)`

4 channel (alpha as a last channel) 16-bit unsigned histogram with bins determined by *pLevels*.

Alpha channel is ignored during the histograms computations.

#### Parameters:

***pSrc*** [Source-Image Pointer](#).

***nSrcStep*** [Source-Image Line Step](#).

***oSizeROI*** [Region-of-Interest \(ROI\)](#).

***pHist*** Array of pointers which are receiving the computed histograms per color channel. Array pointed by *pHist[i]* must be of size *nLevels[i]*-1.

***nLevels*** Array containing number of levels per color channel.

***pLevels*** Array containing pointers to level-arrays per color channel. Array pointed by *pLevel[i]* must be of size *nLevels[i]*.

***pBuffer*** Pointer to appropriately sized (`npPiHistogramRangeGetBufferSize_16u_AC4R`) scratch buffer.

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.119** `NppStatus npPiHistogramRange_16u_C1R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, const Npp32s * pLevels, int nLevels, Npp8u * pBuffer)`

16-bit unsigned histogram with bins determined by *pLevels* array.

#### Parameters:

***pSrc*** [Source-Image Pointer](#).

***nSrcStep*** [Source-Image Line Step](#).

***oSizeROI*** [Region-of-Interest \(ROI\)](#).

***pHist*** Pointer to array that receives the computed histogram. The array must be of size *nLevels*-1.

***pLevels*** Pointer to array containing the level sizes of the bins. The array must be of size *nLevels*.

*nLevels* Number of levels in histogram.

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_16u\_C1R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.120** `NppStatus nppiHistogramRange_16u_C4R (const Npp16u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], const Npp32s * pLevels[4], int nLevels[4], Npp8u * pBuffer)`

4 channel 16-bit unsigned histogram with bins determined by pLevels.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by pHist[i] must be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by pLevel[i] must be of size nLevels[i].

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_16u\_C4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.121** `NppStatus nppiHistogramRange_32f_AC4R (const Npp32f * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[3], const Npp32f * pLevels[3], int nLevels[3], Npp8u * pBuffer)`

4 channel (alpha as a last channel) 32-bit float histogram with bins determined by pLevels.

Alpha channel is ignored during the histograms computations.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by pHist[i] must be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by pLevel[i] must be of size nLevels[i].

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_32f\_AC4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.122 **NppStatus nppiHistogramRange\_32f\_C1R** (const Npp32f \* *pSrc*, int *nSrcStep*, NppiSize *oSizeROI*, Npp32s \* *pHist*, const Npp32f \* *pLevels*, int *nLevels*, Npp8u \* *pBuffer*)

32-bit float histogram with bins determined by *pLevels* array.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Pointer to array that receives the computed histogram. The array must be of size *nLevels*-1.

*pLevels* Pointer to array containing the level sizes of the bins. The array must be of size *nLevels*.

*nLevels* Number of levels in histogram.

*pBuffer* Pointer to appropriately sized (`nppiHistogramRangeGetBufferSize_32f_C1R`) scratch buffer.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.123 **NppStatus nppiHistogramRange\_32f\_C4R** (const Npp32f \* *pSrc*, int *nSrcStep*, NppiSize *oSizeROI*, Npp32s \* *pHist*[4], const Npp32f \* *pLevels*[4], int *nLevels*[4], Npp8u \* *pBuffer*)

4 channel 32-bit float histogram with bins determined by *pLevels*.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by *pHist*[*i*] must be of size *nLevels*[*i*]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by *pLevel*[*i*] must be of size *nLevels*[*i*].

*pBuffer* Pointer to appropriately sized (`nppiHistogramRangeGetBufferSize_32f_C4R`) scratch buffer.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.124 **NppStatus nppiHistogramRange\_8u\_AC4R** (const Npp8u \* *pSrc*, int *nSrcStep*, NppiSize *oSizeROI*, Npp32s \* *pHist*[3], const Npp32s \* *pLevels*[3], int *nLevels*[3], Npp8u \* *pBuffer*)

4 channel (alpha as a last channel) 8-bit unsigned histogram with bins determined by *pLevels*.

Alpha channel is ignored during the histograms computations.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by pHist[i] must be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by pLevel[i] must be of size nLevels[i].

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_8u\_AC4R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.125** `NppStatus nppiHistogramRange_8u_C1R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist, const Npp32s * pLevels, int nLevels, Npp8u * pBuffer)`

8-bit unsigned histogram with bins determined by pLevels array.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Pointer to array that receives the computed histogram. The array must be of size nLevels-1.

*pLevels* Pointer to array containing the level sizes of the bins. The array must be of size nLevels.

*nLevels* Number of levels in histogram.

*pBuffer* Pointer to appropriately sized (nppiHistogramRangeGetBufferSize\_8u\_C1R) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.126** `NppStatus nppiHistogramRange_8u_C4R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pHist[4], const Npp32s * pLevels[4], int nLevels[4], Npp8u * pBuffer)`

4 channel 8-bit unsigned histogram with bins determined by pLevels.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pHist* Array of pointers which are receiving the computed histograms per color channel. Array pointed by pHist[i] must be of size nLevels[i]-1.

*nLevels* Array containing number of levels per color channel.

*pLevels* Array containing pointers to level-arrays per color channel. Array pointed by *pLevel[i]* must be of size *nLevels[i]*.

*pBuffer* Pointer to appropriately sized (`nppiHistogramRangeGetBufferSize_8u_C4R`) scratch buffer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.127 NppStatus nppiHistogramRangeGetBufferSize\_16s\_AC4R (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)**

Scratch-buffer size for `nppiHistogramRange_16s_AC4R`.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.128 NppStatus nppiHistogramRangeGetBufferSize\_16s\_C1R (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)**

Scratch-buffer size for `nppiHistogramRange_16s_C1R`.

**Parameters:**

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nLevels* Number of levels in the histogram.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.129 NppStatus nppiHistogramRangeGetBufferSize\_16s\_C4R (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)**

Scratch-buffer size for `nppiHistogramRange_16s_C4R`.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.130 NppStatus nppiHistogramRangeGetBufferSize\_16u\_AC4R (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramRange\_16u\_AC4R.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.131 NppStatus nppiHistogramRangeGetBufferSize\_16u\_C1R (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramRange\_16u\_C1R.

**Parameters:**

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nLevels* Number of levels in the histogram.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.132 NppStatus nppiHistogramRangeGetBufferSize\_16u\_C4R (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramRange\_16u\_C4R.

**Parameters:**

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.133 NppStatus nppiHistogramRangeGetBufferSize\_32f\_AC4R (NppiSize *oSizeROI*, int *nLevels*[3], int \* *hpBufferSize*)**

Scratch-buffer size for nppiHistogramRange\_32f\_AC4R.

**Parameters:**

*oSizeROI* ROI size.  
*nLevels* Array containing number of levels per color channel.  
*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.134 NppStatus nppiHistogramRangeGetBufferSize\_32f\_C1R (NppiSize oSizeROI, int nLevels, int \* hpBufferSize)**

Scratch-buffer size for nppiHistogramRange\_32f\_C1R.

**Parameters:**

*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*nLevels* Number of levels in the histogram.  
*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.135 NppStatus nppiHistogramRangeGetBufferSize\_32f\_C4R (NppiSize oSizeROI, int nLevels[4], int \* hpBufferSize)**

Scratch-buffer size for nppiHistogramRange\_32f\_C4R.

**Parameters:**

*oSizeROI* ROI size.  
*nLevels* Array containing number of levels per color channel.  
*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.

**7.4.1.136 NppStatus nppiHistogramRangeGetBufferSize\_8u\_AC4R (NppiSize oSizeROI, int nLevels[3], int \* hpBufferSize)**

Scratch-buffer size for nppiHistogramRange\_8u\_AC4R.

**Parameters:**

*oSizeROI* ROI size.  
*nLevels* Array containing number of levels per color channel.  
*hpBufferSize* Host pointer where required buffer size is returned.

**Returns:**

Error Code.



#### 7.4.1.137 NppStatus nppiHistogramRangeGetBufferSize\_8u\_C1R (NppiSize *oSizeROI*, int *nLevels*, int \* *hpBufferSize*)

Scratch-buffer size for nppiHistogramRange\_8u\_C1R.

##### Parameters:

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nLevels* Number of levels in the histogram.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.138 NppStatus nppiHistogramRangeGetBufferSize\_8u\_C4R (NppiSize *oSizeROI*, int *nLevels*[4], int \* *hpBufferSize*)

Scratch-buffer size for nppiHistogramRange\_8u\_C4R.

##### Parameters:

*oSizeROI* ROI size.

*nLevels* Array containing number of levels per color channel.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.139 NppStatus nppiLn\_32f\_C1R (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit floating point logarithm.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.140** `NppStatus nppiLUT_Linear_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp32s * pValues[4], const Npp32s * pLevels[4], int nLevels[4])`

4 channel 8-bit unsigned look-up-table color conversion, not affecting Alpha.

The LUT is derived from a set of user defined mapping points through linear interpolation. Alpha channel is the last channel and is not processed.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pValues* Double pointer to an [4] of arrays of user defined OUTPUT values per CHANNEL

*pLevels* Double pointer to an [4] of arrays of user defined INPUT values per CHANNEL

*nLevels* A [4] array of user defined input/output mapping points (levels) per CHANNEL

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_LUT\\_NUMBER\\_OF\\_LEVELS\\_ERROR](#) if the number of levels is less than 2.

**7.4.1.141** `NppStatus nppiLUT_Linear_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp32s * pValues, const Npp32s * pLevels, int nLevels)`

8-bit unsigned look-up-table color conversion.

The LUT is derived from a set of user defined mapping points through linear interpolation.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pValues* Pointer to an array of user defined OUTPUT values

*pLevels* Pointer to an array of user defined INPUT values

*nLevels* Number of user defined input/output mapping points (levels)

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_LUT\\_NUMBER\\_OF\\_LEVELS\\_ERROR](#) if the number of levels is less than 2.

**7.4.1.142** `NppStatus nppiLUT_Linear_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp32s * pValues[3], const Npp32s * pLevels[3], int nLevels[3])`

3 channel 8-bit unsigned look-up-table color conversion.

The LUT is derived from a set of user defined mapping points through linear interpolation.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pValues* Double pointer to an [3] of arrays of user defined OUTPUT values per CHANNEL

*pLevels* Double pointer to an [3] of arrays of user defined INPUT values per CHANNEL

*nLevels* A [3] array of user defined input/output mapping points (levels) per CHANNEL

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

- `NPP_LUT_NUMBER_OF_LEVELS_ERROR` if the number of levels is less than 2.

**7.4.1.143** `NppStatus nppiMagnitude_32fc32f_C1R (const Npp32fc * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point complex to 32-bit floating point magnitude.

Converts complex-number pixel image to single channel image computing the result pixels as the magnitude of the complex values.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.144** `NppStatus nppiMagnitudeSqr_32fc32f_C1R (const Npp32fc * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point complex to 32-bit floating point squared magnitude.

Converts complex-number pixel image to single channel image computing the result pixels as the squared magnitude of the complex values.

The squared magnitude is an intermediate result of magnitude computation and can thus be computed faster than actual magnitude. If magnitudes are required for sorting/comparing only, using this function instead of `nppiMagnitude_32fc32f_C1R` can be a worthwhile performance optimization.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.145 Npp16s\* nppiMalloc\_16s\_C1 (int nWidthPixels, int nHeightPixels, int \* pStepBytes)**

16-bit signed image memory allocator.

**Parameters:**

*nWidthPixels* Image width.  
*nHeightPixels* Image height.  
*pStepBytes* Line Step.

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.146 Npp16s\* nppiMalloc\_16s\_C4 (int nWidthPixels, int nHeightPixels, int \* pStepBytes)**

4 channel 16-bit signed image memory allocator.

**Parameters:**

*nWidthPixels* Image width.  
*nHeightPixels* Image height.  
*pStepBytes* Line Step.

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.147 Npp16u\* nppiMalloc\_16u\_C1 (int nWidthPixels, int nHeightPixels, int \* pStepBytes)**

16-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.148 Npp16u\* nppiMalloc\_16u\_C3 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

3 channel 16-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.149 Npp16u\* nppiMalloc\_16u\_C4 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

4 channel 16-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.150 Npp32f\* nppiMalloc\_32f\_C1 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

32-bit floating point image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.151 Npp32f\* nppiMalloc\_32f\_C2 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

2 channel 32-bit floating point image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.152 Npp32f\* nppiMalloc\_32f\_C3 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

3 channel 32-bit floating point image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.153 Npp32f\* nppiMalloc\_32f\_C4 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

4 channel 32-bit floating point image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.154 Npp32s\* nppiMalloc\_32s\_C1 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

32-bit signed image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.155 Npp32s\* nppiMalloc\_32s\_C3 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

3 channel 32-bit signed image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.156 Npp32s\* nppiMalloc\_32s\_C4 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

4 channel 32-bit signed image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.157 Npp8u\* nppiMalloc\_8u\_C1 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

8-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.158 Npp8u\* nppiMalloc\_8u\_C2 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

2 channel 8-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.159 Npp8u\* nppiMalloc\_8u\_C3 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

3 channel 8-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.160 Npp8u\* nppiMalloc\_8u\_C4 (int *nWidthPixels*, int *nHeightPixels*, int \* *pStepBytes*)**

4 channel 8-bit unsigned image memory allocator.

**Parameters:**

*nWidthPixels* Image width.

*nHeightPixels* Image height.

*pStepBytes* [Line Step](#).

**Returns:**

Pointer to new image data. 0 (null-pointer) indicates allocation failure.

**7.4.1.161 NppStatus nppiMean\_StdDev\_8u\_C1R (const Npp8u \* *pSrc*, int *nSrcStep*, NppiSize *oSizeROI*, Npp64f \* *pMean*, Npp64f \* *pStdDev*)**

8-bit unsigned mean standard deviation.

**Parameters:**

*pSrc* [Source-Image Pointer](#).



*nSrcStep* Source-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMean* Contains computed mean. This is a host pointer.  
*pStdDev* Contains computed standard deviation. This is a host pointer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.162 NppStatus nppiMinMax\_8u\_C1R (const Npp8u \* pSrc, int nSrcStep, NppiSize oSizeROI, Npp8u \* pMin, Npp8u \* pMax)

8-bit unsigned pixel minimum and maximum.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMin* Host-memory pointer receiving the minimum result.  
*pMax* Host-memory pointer receiving the maximum result.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**Note:**

This primitive returns the min/max results in HOST POINTERS.

#### 7.4.1.163 NppStatus nppiMinMax\_8u\_C4R (const Npp8u \* pSrc, int nSrcStep, NppiSize oSizeROI, Npp8u cuMin[4], Npp8u cuMax[4])

4 channel 8-bit unsigned pixel minimum and maximum.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*cuMin* Device-pointer (array) receiving the minimum result.  
*cuMax* Device-pointer (array) receiving the maximum result.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**Note:**

Unlike nppiMinMax\_8u\_C1R, this primitive returns its results as device pointers.

#### 7.4.1.164 NppStatus nppiMirror\_8u\_C1R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oROI*, NppiAxis *flip*)

8-bit unsigned image mirror.

##### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oROI* Region-of-Interest (ROI).

*flip* Specifies the axis about which the image is to be mirrored.

##### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- **NPP\_MIRROR\_FLIP\_ERR** if flip has an illegal value.

#### 7.4.1.165 NppStatus nppiMirror\_8u\_C4R (const Npp8u \* *pSrc*, int *nSrcStep*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oROI*, NppiAxis *flip*)

4 channel 8-bit unsigned image mirror.

##### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Distance in bytes between starts of consecutive lines of the destination image.

*oROI* Region-of-Interest (ROI).

*flip* Specifies the axis about which the image is to be mirrored.

##### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- **NPP\_MIRROR\_FLIP\_ERR** if flip has an illegal value.

#### 7.4.1.166 NppStatus nppiMul\_32f\_C1R (const Npp32f \* *pSrc1*, int *nSrc1Step*, const Npp32f \* *pSrc2*, int *nSrc2Step*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 32-bit floating point image multiplication.

Multiply corresponding pixels in ROI.

##### Parameters:

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.167** `NppStatus nppiMul_32s_C1R (const Npp32s * pSrc1, int nSrc1Step, const Npp32s * pSrc2, int nSrc2Step, Npp32s * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 32-bit image multiplication.

Multiply corresponding pixels in ROI.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.168** `NppStatus nppiMul_8u_AC4RSfs (const Npp8u * pSrc1, int nSrc1Step, const Npp8u * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, int nScaleFactor)`

4 channel 8-bit unsigned image multiplication, not affecting Alpha.

Multiply corresponding pixels in ROI.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.169 **NppStatus nppiMul\_8u\_C1RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

8-bit unsigned image multiplication.

Multiply the pixel values of corresponding pixels in the ROI and write them to the output image.

##### Parameters:

*pSrc1* [Source-Image Pointer](#).

*nSrc1Step* [Source-Image Line Step](#).

*pSrc2* [Source-Image Pointer](#).

*nSrc2Step* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.170 **NppStatus nppiMul\_8u\_C4RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

4 channel 8-bit unsigned image multiplication.

Multiply corresponding pixels in ROI.

##### Parameters:

*pSrc1* [Source-Image Pointer](#).

*nSrc1Step* [Source-Image Line Step](#).

*pSrc2* [Source-Image Pointer](#).

*nSrc2Step* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.171 **NppStatus nppiMulC\_32f\_C1R** (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f *nValue*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit floating point image multiply constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.172** `NppStatus nppiMulC_32fc_C1R (const Npp32fc * pSrc, int nSrcStep, Npp32fc nValue, Npp32fc * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit complex floating point image multiply constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.173** `NppStatus nppiNormDiff_Inf_8u_C1R (const Npp8u * pSrc1, int nSrcStep1, const Npp8u * pSrc2, int nSrcStep2, NppiSize oSizeROI, Npp64f * pRetVal)`

8-bit unsigned Infinity Norm of pixel differences.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrcStep1* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrcStep2* Source-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*\*pRetVal* Contains computed L1-norm of differences. This is a host pointer.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.174** `NppStatus nppiNormDiff_L1_8u_C1R (const Npp8u * pSrc1, int nSrcStep1, const Npp8u * pSrc2, int nSrcStep2, NppiSize oSizeROI, Npp64f * pRetVal)`

8-bit unsigned L1 norm of pixel differences.

**Parameters:**

*pSrc1* Source-Image Pointer.

*nSrcStep1* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrcStep2* Source-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pRetVal* Contains computed L1-norm of differences. This is a host pointer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.175** `NppStatus nppiNormDiff_L2_8u_C1R (const Npp8u * pSrc1, int nSrcStep1, const Npp8u * pSrc2, int nSrcStep2, NppiSize oSizeROI, Npp64f * pRetVal)`

8-bit unsigned L2 norm of pixel differences.

**Parameters:**

*pSrc1* Source-Image Pointer.

*nSrcStep1* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrcStep2* Source-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pRetVal* Contains computed L1-norm of differences. This is a host pointer.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.176** `NppStatus nppiQuantFwdRawTableInit_JPEG_8u (Npp8u * pQuantRawTable, int nQualityFactor)`

Converts regular quantization tables with the quality factor.

**Parameters:**

*pQuantRawTable* Raw quantization table.

*nQualityFactor* Quality factor for the table. Range is [1:100].

**Returns:**

[NPP\\_NULL\\_POINTER\\_ERROR](#) pQuantRawTable is a null pointer.

#### 7.4.1.177 `NppStatus nppiQuantFwdTableInit_JPEG_8u16u (const Npp8u * pQuantRawTable, Npp16u * pQuantFwdRawTable)`

Converts raw quantization table to a forward quantization table.

##### Parameters:

*pQuantRawTable* Raw quantization table.

*pQuantFwdRawTable* Forward quantization table.

##### Returns:

[NPP\\_NULL\\_POINTER\\_ERROR](#) *pQuantRawTable* is a null pointer.

#### 7.4.1.178 `NppStatus nppiQuantInvTableInit_JPEG_8u16u (const Npp8u * pQuantRawTable, Npp16u * pQuantFwdRawTable)`

Converts raw quantization table to an inverse quantization table.

##### Parameters:

*pQuantRawTable* Raw quantization table.

*pQuantFwdRawTable* Inverse quantization table.

##### Returns:

[NPP\\_NULL\\_POINTER\\_ERROR](#) *pQuantRawTable* or *pQuantFwdRawTable* is a null pointer.

#### 7.4.1.179 `NppStatus nppiRectStdDev_32s32f_C1R (const Npp32s * pSrc, int nSrcStep, const Npp32f * pSqr, int nSqrStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, NppiRect rect)`

*RectStdDev* Computes the standard deviation of integral images.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pSqr* [Destination-Image Pointer](#).

*nSqrStep* [Destination-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rect* [rectangular window](#)

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.180 **NppStatus nppiReductionGetBufferHostSize\_8u\_C1R** (const NppiSize & *oSizeROI*, int \* *hpBufferSize*)

Scratch-buffer size for nppiSum\_8u\_C1R.

##### Parameters:

*oSizeROI* ROI size.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.181 **NppStatus nppiReductionGetBufferHostSize\_8u\_C4R** (const NppiSize & *oSizeROI*, int \* *hpBufferSize*)

Scratch-buffer size for nppiSum\_8u\_C4R.

##### Parameters:

*oSizeROI* ROI size.

*hpBufferSize* Host pointer where required buffer size is returned.

##### Returns:

Error Code.

#### 7.4.1.182 **NppStatus nppiResize\_8u\_C1R** (const Npp8u \* *pSrc*, NppiSize *srcSize*, int *nSrcStep*, NppiRect *srcROI*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *dstROISize*, double *xFactor*, double *yFactor*, int *interpolation*)

8-bit unsigned image resize.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*srcSize* Size in pixels of the source image

*srcROI* Region of interest in the source image.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstROISize* Size in pixels of the destination image

*xFactor* Factors by which x dimension is changed

*yFactor* Factors by which y dimension is changed

*interpolation* The type of interpolation to perform resampling

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)



- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) indicates an error condition if srcROIRect has no intersection with the source image.
- [NPP\\_RESIZE\\_NO\\_OPERATION\\_ERROR](#) if either destination ROI width or height is less than 1 pixel.
- [NPP\\_RESIZE\\_FACTOR\\_ERROR](#) Indicates an error condition if either xFactor or yFactor is less than or equal to zero.
- [NPP\\_INTERPOLATION\\_ERROR](#) if interpolation has an illegal value.

**7.4.1.183** `NppStatus nppiResize_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcROI, Npp8u * pDst, int nDstStep, NppiSize dstROISize, double xFactor, double yFactor, int interpolation)`

4 channel 8-bit unsigned image resize.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*srcSize* Size in pixels of the source image

*srcROI* Region of interest in the source image.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstROISize* Size in pixels of the destination image

*xFactor* Factors by which x dimension is changed

*yFactor* Factors by which y dimension is changed

*interpolation* The type of interpolation to perform resampling

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) indicates an error condition if srcROIRect has no intersection with the source image.
- [NPP\\_RESIZE\\_NO\\_OPERATION\\_ERROR](#) if either destination ROI width or height is less than 1 pixel.
- [NPP\\_RESIZE\\_FACTOR\\_ERROR](#) Indicates an error condition if either xFactor or yFactor is less than or equal to zero.
- [NPP\\_INTERPOLATION\\_ERROR](#) if interpolation has an illegal value.

**7.4.1.184** `NppStatus nppiRGBToYCbCr420_8u_C3P3R (const Npp8u * pSrc, int nSrcStep, Npp8u ** pDst, int nDstStep[3], NppiSize oSizeROI)`

3 channel 8-bit unsigned packed RGB to planar YCbCr420 color conversion.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.185 NppStatus nppiRGBToYCbCr422\_8u\_C3C2R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

3 channel 8-bit unsigned RGB to 2 channel chroma packed YCbCr422 color conversion.  
images.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.186 NppStatus nppiRGBToYCbCr\_8u\_AC4R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 8-bit unsigned RGB to YCbCr color conversion, ignoring Alpha.  
Alpha channel is the last channel and is not processed.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.187** `NppStatus nppiRGBToYCbCr_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

3 channel 8-bit unsigned packed RGB to packed YCbCr color conversion.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.188** `NppStatus nppiRGBToYCbCr_8u_P3R (const Npp8u *const * pSrc, int nSrcStep, Npp8u ** pDst, int nDstStep, NppiSize oSizeROI)`

3 channel planar 8-bit unsigned RGB to YCbCr color conversion.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.189** `NppStatus nppiRotate_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcROI, Npp8u * pDst, int nDstStep, NppiRect dstROI, double angle, double xShift, double yShift, int interpolation)`

8-bit unsigned image rotate.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*srcSize* Size in pixels of the source image  
*srcROI* Region of interest in the source image.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.

*dstROI* Region of interest in the destination image.

*angle* The angle of rotation in degrees.

*xShift* Shift along horizontal axis

*yShift* Shift along vertical axis

*interpolation* The type of interpolation to perform resampling

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_INTERPOLATION\\_ERROR](#) if interpolation has an illegal value.
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1.
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) indicates an error condition if srcROIrect has no intersection with the source image.
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) indicates a warning that no operation is performed if the transformed source ROI does not intersect the destination ROI.

**7.4.1.190** `NppStatus nppiRotate_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcROI, Npp8u * pDst, int nDstStep, NppiRect dstROI, double angle, double xShift, double yShift, int interpolation)`

4 channel 8-bit unsigned image rotate.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*srcSize* Size in pixels of the source image

*srcROI* Region of interest in the source image.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstROI* Region of interest in the destination image.

*angle* The angle of rotation in degrees.

*xShift* Shift along horizontal axis

*yShift* Shift along vertical axis

*interpolation* The type of interpolation to perform resampling

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_INTERPOLATION\\_ERROR](#) if interpolation has an illegal value.
- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1.
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) indicates an error condition if srcROIrect has no intersection with the source image.
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) indicates a warning that no operation is performed if the transformed source ROI does not intersect the destination ROI.

#### 7.4.1.191 **NppStatus nppiSet\_16s\_AC4MR** (const Npp16s *aValues*[3], Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp8u \* *pMask*, int *nMaskStep*)

Masked 4 channel 16-bit image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.192 **NppStatus nppiSet\_16s\_AC4R** (const Npp16s *aValues*[3], Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

4 channel 16-bit image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.193 **NppStatus nppiSet\_16s\_C1MR** (Npp16s *nValue*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp8u \* *pMask*, int *nMaskStep*)

Masked 16-bit image set.

##### Parameters:

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.194 NppStatus nppiSet\_16s\_C1R (Npp16s nValue, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI)**

16-bit image set.

**Parameters:**

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.195 NppStatus nppiSet\_16s\_C2R (const Npp16s aValues[2], Npp16s \* pDst, int nDstStep, NppiSize oSizeROI)**

2 channel 16-bit image set.

**Parameters:**

*aValues* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.196 NppStatus nppiSet\_16s\_C4CR (Npp16s nValue, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 16-bit unsigned image set affecting only single channel.

For RGBA images, this method allows setting of a single of the four (RGBA) values without changing the contents of the other three channels. The channel is selected via the pDst pointer. The pointer needs to point to the actual first value to be set, e.g. in order to set the R-channel (first channel), one would pass pDst unmodified, since its value actually points to the r channel. If one wanted to modify the B channel (second channel), one would pass pDst + 2 to the function.

**Parameters:**

*nValue* The pixel-value to be set.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.197** `NppStatus nppiSet_16s_C4MR (const Npp16s aValues[4], Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 16-bit image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.  
*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.198** `NppStatus nppiSet_16s_C4R (const Npp16s aValues[4], Npp16s * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 16-bit image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.199 `NppStatus nppiSet_16u_AC4MR (const Npp16u aValues[3], Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 16-bit unsigned image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.200 `NppStatus nppiSet_16u_AC4R (const Npp16u aValues[3], Npp16u * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 16-bit unsigned image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.201 `NppStatus nppiSet_16u_C1MR (Npp16u nValue, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 16-bit unsigned image set.

##### Parameters:

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).



*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.202 NppStatus nppiSet\_16u\_C1R (Npp16u nValue, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI)**

16-bit unsigned image set.

**Parameters:**

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.203 NppStatus nppiSet\_16u\_C2R (const Npp16u aValues[2], Npp16u \* pDst, int nDstStep, NppiSize oSizeROI)**

2 channel 16-bit unsigned image set.

**Parameters:**

*aValues* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.204 NppStatus nppiSet\_16u\_C4CR (Npp16u nValue, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 16-bit unsigned image set affecting only single channel.

For RGBA images, this method allows setting of a single of the four (RGBA) values without changing the contents of the other three channels. The channel is selected via the pDst pointer. The pointer needs to point to the actual first value to be set, e.g. in order to set the R-channel (first channel), one would pass pDst unmodified, since its value actually points to the r channel. If one wanted to modify the B channel (second channel), one would pass pDst + 2 to the function.

**Parameters:**

*nValue* The pixel-value to be set.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.205** `NppStatus nppiSet_16u_C4MR (const Npp16u aValues[4], Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 16-bit unsigned image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.  
*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.206** `NppStatus nppiSet_16u_C4R (const Npp16u aValues[4], Npp16u * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 16-bit unsigned image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.207 `NppStatus nppiSet_32f_AC4MR (const Npp32f aValues[3], Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 32-bit floating point image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.208 `NppStatus nppiSet_32f_AC4R (const Npp32f aValues[3], Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 32-bit floating point image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.209 `NppStatus nppiSet_32f_C1MR (Npp32f nValue, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 32-bit floating point image set.

##### Parameters:

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.210 NppStatus nppiSet\_32f\_C1R (Npp32f nValue, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI)**

32-bit floating point image set.

**Parameters:**

*nValue* New pixel value.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.211 NppStatus nppiSet\_32f\_C4CR (Npp32f nValue, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI)**

4 channel 32-bit floating point image set affecting only single channel.

For RGBA images, this method allows setting of a single of the four (RGBA) values without changing the contents of the other three channels. The channel is selected via the pDst pointer. The pointer needs to point to the actual first value to be set, e.g. in order to set the R-channel (first channel), one would pass pDst unmodified, since its value actually points to the r channel. If one wanted to modify the B channel (second channel), one would pass pDst + 2 to the function.

**Parameters:**

*nValue* The pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.212 NppStatus nppiSet\_32f\_C4MR (const Npp32f aValues[4], Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u \* pMask, int nMaskStep)**

Masked 4 channel 32-bit floating point image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.  
*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.213 NppStatus nppiSet\_32f\_C4R (const Npp32f aValues[4], Npp32f \* pDst, int nDstStep, NppiSize oSizeROI)

4 channel 32-bit floating point image set.

**Parameters:**

*aValues* New pixel value.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.214 NppStatus nppiSet\_32s\_AC4MR (const Npp32s aValues[3], Npp32s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u \* pMask, int nMaskStep)

Masked 4 channel 16-bit image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

**Parameters:**

*aValues* Three-channel array containing the pixel-value to be set.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.  
*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.215 `NppStatus nppiSet_32s_AC4R (const Npp32s aValues[3], Npp32s * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 16-bit image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

##### Parameters:

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.216 `NppStatus nppiSet_32s_C1MR (Npp32s nValue, Npp32s * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 32-bit image set.

##### Parameters:

*nValue* New pixel value.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.217 `NppStatus nppiSet_32s_C1R (Npp32s nValue, Npp32s * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit image set.

##### Parameters:

*nValue* New pixel value.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.218 `NppStatus nppiSet_32s_C4CR (Npp32s nValue, Npp32s * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 32-bit unsigned image set affecting only single channel.

For RGBA images, this method allows setting of a single of the four (RGBA) values without changing the contents of the other three channels. The channel is selected via the *pDst* pointer. The pointer needs to point to the actual first value to be set, e.g. in order to set the R-channel (first channel), one would pass *pDst* unmodified, since its value actually points to the r channel. If one wanted to modify the B channel (second channel), one would pass *pDst* + 2 to the function.

##### Parameters:

*nValue* The pixel-value to be set.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.219 `NppStatus nppiSet_32s_C4MR (const Npp32s aValues[4], Npp32s * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 32-bit image set.

##### Parameters:

*aValues* New pixel value.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.220 `NppStatus nppiSet_32s_C4R (const Npp32s aValues[4], Npp32s * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 32-bit image set.

##### Parameters:

*aValues* New pixel value.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.221** `NppStatus nppiSet_8u_AC4MR (const Npp8u aValues[3], Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 8-bit unsigned image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

**Parameters:**

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.222** `NppStatus nppiSet_8u_AC4R (const Npp8u aValues[3], Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 8-bit unsigned image set method, not affecting Alpha channel.

For RGBA images, this method allows setting of the RGB values without changing the contents of the alpha-channel (fourth channel).

**Parameters:**

*aValues* Three-channel array containing the pixel-value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)



#### 7.4.1.223 `NppStatus nppiSet_8u_C1MR (Npp8u nValue, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 8-bit unsigned image set.

The 8-bit mask image affects setting of the respective pixels in the destination image. If the mask value is zero (0) the pixel is not set, if the mask is non-zero, the corresponding destination pixel is set to specified value.

##### Parameters:

*nValue* The pixel value to be set.

*pDst* Pointer [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.224 `NppStatus nppiSet_8u_C1R (Npp8u nValue, Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

8-bit unsigned image set.

##### Parameters:

*nValue* The pixel value to be set.

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.225 `NppStatus nppiSet_8u_C4CR (Npp8u nValue, Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 8-bit unsigned image set affecting only single channel.

For RGBA images, this method allows setting of a single of the four (RGBA) values without changing the contents of the other three channels. The channel is selected via the *pDst* pointer. The pointer needs to point to the actual first value to be set, e.g. in order to set the R-channel (first channel), one would pass *pDst* unmodified, since its value actually points to the r channel. If one wanted to modify the B channel (second channel), one would pass *pDst* + 2 to the function.

##### Parameters:

*nValue* The pixel-value to be set.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.226** `NppStatus nppiSet_8u_C4MR (const Npp8u aValues[4], Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u * pMask, int nMaskStep)`

Masked 4 channel 8-bit unsigned image set.

**Parameters:**

*aValues* Four-channel array containing the pixel-value to be set.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pMask* Pointer to the mask image. This is a single channel 8-bit unsigned int image.

*nMaskStep* Number of bytes between line starts of successive lines in the mask image.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.227** `NppStatus nppiSet_8u_C4R (const Npp8u aValues[4], Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 8-bit unsigned image set.

**Parameters:**

*aValues* Four-channel array containing the pixel-value to be set.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.228** `NppStatus nppiSetDefaultQuantTable (Npp8u * pQuantRawTable, int tableIndex)`

Fills out the quantization table with either luminance and chrominance tables for JPEG.

**Parameters:**

*pQuantRawTable* Raw quantization table.

*tableIndex* Choice for Luminance (tableIndex is 0) or Chrominance component (tableIndex is 1).

**Returns:**

Error codes:

- [NPP\\_NULL\\_POINTER\\_ERROR](#) pQuantRawTable is a null pointer.
- [NPP\\_INVALID\\_INPUT](#) if tableIndex is not 0 or 1.

**7.4.1.229** `NppStatus nppiSqrIntegral_8u32s32f_C1R (Npp8u * pSrc, int nSrcStep, Npp32s * pDst, int nDstStep, Npp32f * pSqr, int nSqrStep, NppiSize srcROI, Npp32s val, Npp32f valSqr, Npp32s integralImageNewHeight)`

SqrIntegral Transforms an image to integral and integral of pixel squares representation.

This function assumes that the integral and integral of squares images.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*pSqr* Destination-Image Pointer.

*nSqrStep* Destination-Image Line Step.

*srcROI* Region-of-Interest (ROI).

*val* The value to add to pDst image pixels

*valSqr* The value to add to pSqr image pixels

*integralImageNewHeight* Extended height of output surfaces (needed by transpose in primitive)

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.230** `NppStatus nppiSub_32f_C1R (const Npp32f * pSrc1, int nSrc1Step, const Npp32f * pSrc2, int nSrc2Step, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point image subtraction.

Subtract pSrc1's pixels from corresponding pixels in pSrc2.

**Parameters:**

*pSrc1* Source-Image Pointer.

*nSrc1Step* Source-Image Line Step.

*pSrc2* Source-Image Pointer.

*nSrc2Step* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

#### 7.4.1.231 **NppStatus nppiSub\_32s\_C1R** (const Npp32s \* *pSrc1*, int *nSrc1Step*, const Npp32s \* *pSrc2*, int *nSrc2Step*, Npp32s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*)

32-bit image subtraction.

Subtract pSrc1's pixels from corresponding pixels in pSrc2.

##### Parameters:

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

##### Returns:

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.232 **NppStatus nppiSub\_8u\_AC4RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

4 channel 8-bit unsigned image subtraction, not affecting Alpha.

Subtract pSrc1's pixels from corresponding pixels in pSrc2.

##### Parameters:

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

##### Returns:

Image Data Related Error Codes, ROI Related Error Codes

#### 7.4.1.233 **NppStatus nppiSub\_8u\_C1RSfs** (const Npp8u \* *pSrc1*, int *nSrc1Step*, const Npp8u \* *pSrc2*, int *nSrc2Step*, Npp8u \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, int *nScaleFactor*)

8-bit unsigned image subtraction.

Subtract the pixel values of corresponding pixels in the ROI and write them to the output image.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.234** `NppStatus nppiSub_8u_C4RSfs (const Npp8u * pSrc1, int nSrc1Step, const Npp8u * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, int nScaleFactor)`

4 channel 8-bit unsigned image subtraction.

Subtract pSrc1's pixels from corresponding pixels in pSrc2.

**Parameters:**

*pSrc1* Source-Image Pointer.  
*nSrc1Step* Source-Image Line Step.  
*pSrc2* Source-Image Pointer.  
*nSrc2Step* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nScaleFactor* Result pixel values are scaled by  $2^{(-nScaleFactor)}$  and then clamped to [0,255] range.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.235** `NppStatus nppiSubC_32f_C1R (const Npp32f * pSrc, int nSrcStep, Npp32f nValue, Npp32f * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit floating point image subtract constant.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.236** `NppStatus nppiSubC_32fc_C1R (const Npp32fc * pSrc, int nSrcStep, Npp32fc nValue, Npp32fc * pDst, int nDstStep, NppiSize oSizeROI)`

32-bit complex floating point image subtract constant.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nValue* Constant.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.237** `NppStatus nppiSum_8u_C1R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pDeviceBuffer, Npp64f * pSum)`

8-bit unsigned image sum.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pDeviceBuffer* Pointer to the required device memory allocation.

*\*pSum* Contains computed sum. This is a host pointer.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.238** `NppStatus nppiSum_8u_C4R (const Npp8u * pSrc, int nSrcStep, NppiSize oSizeROI, Npp32s * pDeviceBuffer, Npp64f aSum[4])`

4 channel 8-bit unsigned image sum.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*pDeviceBuffer* Pointer to the required device memory allocation.

*aSum* Array contains computed sum for each channel. This is a host pointer.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.239** `NppStatus nppiSumWindowColumn_8u32f_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp32f * pDst, Npp32s nDstStep, NppiSize oROI, Npp32s nMaskSize, Npp32s nAnchor)`

8-bit unsigned 1D (column) sum to 32f.

Apply Column Window Summation filter over a 1D mask region around each source pixel for 1-channel 8 bit/pixel input images with 32-bit floating point output. Result 32-bit floating point pixel is equal to the sum of the corresponding and neighboring column pixel values in a mask region of the source image defined by nMaskSize and nAnchor.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oROI* Region-of-Interest (ROI).

*nMaskSize* Length of the linear kernel array.

*nAnchor* Y offset of the kernel origin frame of reference w.r.t the source pixel.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.240** `NppStatus nppiSumWindowRow_8u32f_C1R (const Npp8u * pSrc, Npp32s nSrcStep, Npp32f * pDst, Npp32s nDstStep, NppiSize oROI, Npp32s nMaskSize, Npp32s nAnchor)`

8-bit unsigned 1D (row) sum to 32f.

Apply Row Window Summation filter over a 1D mask region around each source pixel for 1-channel 8-bit pixel input images with 32-bit floating point output. Result 32-bit floating point pixel is equal to the sum of the corresponding and neighboring row pixel values in a mask region of the source image defined by iKernelDim and iAnchorX.

#### Parameters:

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oROI* Region-of-Interest (ROI).

*nMaskSize* Length of the linear kernel array.

*nAnchor* X offset of the kernel origin frame of reference w.r.t the source pixel.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.241 NppStatus nppiSwapChannels\_8u\_C4IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const int aDstOrder[4])**

4 channel 8-bit unsigned swap channels, in-place.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place Line Step.

*oSizeROI* Region-of-Interest (ROI).

*aDstOrder* Integer array describing how channel values are permuted. The n-th entry of the array contains the number of the channel that is stored in the n-th channel of the output image. E.g. Given an RGBA image, aDstOrder = [3,2,1,0] converts this to ABGR channel order.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.242 NppStatus nppiThreshold\_32f\_C1R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, Npp32f nThreshold, NppCmpOp eComparisonOperation)**

32-bit floating point threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

*nThreshold* The threshold value.

*eComparisonOperation* The type of comparison operation to be used. The only valid values are: NPP\_CMP\_LESS and NPP\_CMP\_GREATER.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#), or NPP\_NOT\_SUPPORTED\_MODE\_ERROR if an invalid comparison operation type is specified.



**7.4.1.243** `NppStatus nppiThreshold_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u aThresholds[3], NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned image threshold, not affecting Alpha.

If for a comparison operations OP the predicate (sourcePixel.channel OP nThreshold) is true, the channel value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*aThresholds* The threshold values, one per color channel.

*eComparisonOperation* The type of comparison operation to be used. The only valid values are: NPP\_CMP\_LESS and NPP\_CMP\_GREATER.

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#), or NPP\_NOT\_SUPPORTED\_MODE\_ERROR if an invalid comparison operation type is specified.

**7.4.1.244** `NppStatus nppiTranspose_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oROI)`

8-bit image transpose.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* Pointer to the destination ROI.

*nDstStep* [Destination-Image Line Step](#).

*oROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.245** `NppStatus nppiWarpAffine_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (16bit unsigned integer, four channels RGBA).

**See also:**

[nppiWarpAffine\\_16u\\_C1R](#)

#### 7.4.1.246 **NppStatus nppiWarpAffine\_16u\_C1R** (const Npp16u \* *pSrc*, NppiSize *srcSize*, int *nSrcStep*, NppiRect *srcRoi*, Npp16u \* *pDst*, int *nDstStep*, NppiRect *dstRoi*, const double *coeffs*[2][3], int *interpolation*)

Affine transform of an image (16bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = C_{00} * x + C_{01} * y + C_{02} \quad Y_{new} = C_{10} * x + C_{11} * y + C_{12}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.247** `NppStatus nppiWarpAffine_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (16bit unsigned integer, three channels).

See also:

[nppiWarpAffine\\_16u\\_C1R](#)

**7.4.1.248** `NppStatus nppiWarpAffine_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (16bit unsigned integer, four channels).

See also:

[nppiWarpAffine\\_16u\\_C1R](#)

**7.4.1.249** `NppStatus nppiWarpAffine_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpAffine\\_16u\\_C1R](#)

**7.4.1.250** `NppStatus nppiWarpAffine_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpAffine\\_16u\\_C1R](#)

**7.4.1.251** `NppStatus nppiWarpAffine_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpAffine\\_32f\\_C1R](#)

**7.4.1.252** `NppStatus nppiWarpAffine_32f_C1R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = C_{00} * x + C_{01} * y + C_{02} \quad Y_{new} = C_{10} * x + C_{11} * y + C_{12}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.253** `NppStatus nppiWarpAffine_32f_C3R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, three channels).

See also:

[nppiWarpAffine\\_32f\\_C1R](#)

**7.4.1.254** `NppStatus nppiWarpAffine_32f_C4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, four channels).

See also:

[nppiWarpAffine\\_32f\\_C1R](#)

**7.4.1.255** `NppStatus nppiWarpAffine_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, three planes).

See also:

[nppiWarpAffine\\_32f\\_C1R](#)

**7.4.1.256** `NppStatus nppiWarpAffine_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit float, four planes).

See also:

[nppiWarpAffine\\_32f\\_C1R](#)

**7.4.1.257** `NppStatus nppiWarpAffine_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpAffine\\_32s\\_C1R](#)

**7.4.1.258** `NppStatus nppiWarpAffine_32s_C1R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = C_{00} * x + C_{01} * y + C_{02} \quad Y_{new} = C_{10} * x + C_{11} * y + C_{12}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid

- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.259** `NppStatus nppiWarpAffine_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, three channels).

See also:

[nppiWarpAffine\\_32s\\_C1R](#)

**7.4.1.260** `NppStatus nppiWarpAffine_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, four channels).

See also:

[nppiWarpAffine\\_32s\\_C1R](#)

**7.4.1.261** `NppStatus nppiWarpAffine_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpAffine\\_32s\\_C1R](#)

**7.4.1.262** `NppStatus nppiWarpAffine_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpAffine\\_32s\\_C1R](#)

**7.4.1.263** `NppStatus nppiWarpAffine_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpAffine\\_8u\\_C1R](#)

**7.4.1.264** `NppStatus nppiWarpAffine_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = C_{00} * x + C_{01} * y + C_{02} \quad Y_{new} = C_{10} * x + C_{11} * y + C_{12}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`



**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.265** `NppStatus nppiWarpAffine_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, three channels).

**See also:**

[nppiWarpAffine\\_8u\\_C1R](#)

**7.4.1.266** `NppStatus nppiWarpAffine_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, four channels).

**See also:**

[nppiWarpAffine\\_8u\\_C1R](#)

**7.4.1.267** `NppStatus nppiWarpAffine_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, three planes).

**See also:**

[nppiWarpAffine\\_8u\\_C1R](#)

**7.4.1.268** `NppStatus nppiWarpAffine_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Affine transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpAffine\\_8u\\_C1R](#)

**7.4.1.269** `NppStatus nppiWarpAffineBack_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, four channels RGBA).

See also:

[nppiWarpAffineBack\\_16u\\_C1R](#)

**7.4.1.270** `NppStatus nppiWarpAffineBack_16u_C1R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpAffineBack`. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$C_{00} * X_{new} + C_{01} * Y_{new} + C_{02} = x \quad C_{10} * X_{new} + C_{11} * Y_{new} + C_{12} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but doesn't perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* Source-Image Line Step.

*srcRoi* Source ROI

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if *interpolation* has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.271** `NppStatus nppiWarpAffineBack_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, three channels).

#### See also:

[nppiWarpAffineBack\\_16u\\_C1R](#)

**7.4.1.272** `NppStatus nppiWarpAffineBack_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, four channels).

#### See also:

[nppiWarpAffineBack\\_16u\\_C1R](#)

**7.4.1.273** `NppStatus nppiWarpAffineBack_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpAffineBack\\_16u\\_C1R](#)

**7.4.1.274** `NppStatus nppiWarpAffineBack_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpAffineBack\\_16u\\_C1R](#)

**7.4.1.275** `NppStatus nppiWarpAffineBack_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpAffineBack\\_32f\\_C1R](#)

**7.4.1.276** `NppStatus nppiWarpAffineBack_32f_C1R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpAffineBack`. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$C_{00} * X_{new} + C_{01} * Y_{new} + C_{02} = x \quad C_{10} * X_{new} + C_{11} * Y_{new} + C_{12} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.277** `NppStatus nppiWarpAffineBack_32f_C3R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, three channels).

#### See also:

[nppiWarpAffineBack\\_32f\\_C1R](#)

**7.4.1.278** `NppStatus nppiWarpAffineBack_32f_C4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, four channels).

#### See also:

[nppiWarpAffineBack\\_32f\\_C1R](#)

**7.4.1.279** `NppStatus nppiWarpAffineBack_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, three planes).

See also:

[nppiWarpAffineBack\\_32f\\_C1R](#)

**7.4.1.280** `NppStatus nppiWarpAffineBack_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit float, four planes).

See also:

[nppiWarpAffineBack\\_32f\\_C1R](#)

**7.4.1.281** `NppStatus nppiWarpAffineBack_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpAffineBack\\_32s\\_C1R](#)

**7.4.1.282** `NppStatus nppiWarpAffineBack_32s_C1R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpAffineBack`. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$C_{00} * X_{new} + C_{01} * Y_{new} + C_{02} = x \quad C_{10} * X_{new} + C_{11} * Y_{new} + C_{12} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.283** `NppStatus nppiWarpAffineBack_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, three channels).

#### See also:

[nppiWarpAffineBack\\_32s\\_C1R](#)

**7.4.1.284** `NppStatus nppiWarpAffineBack_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, four channels).

#### See also:

[nppiWarpAffineBack\\_32s\\_C1R](#)

**7.4.1.285** `NppStatus nppiWarpAffineBack_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpAffineBack\\_32s\\_C1R](#)

**7.4.1.286** `NppStatus nppiWarpAffineBack_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpAffineBack\\_32s\\_C1R](#)

**7.4.1.287** `NppStatus nppiWarpAffineBack_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpAffineBack\\_8u\\_C1R](#)

**7.4.1.288** `NppStatus nppiWarpAffineBack_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetAffineTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpAffineBack`. The function operates on source and destination regions of interest. The affine warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$C_{00} * X_{new} + C_{01} * Y_{new} + C_{02} = x \quad C_{10} * X_{new} + C_{11} * Y_{new} + C_{12} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetAffineQuad` and `nppiGetAffineBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but doesn't perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto



image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Affine transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.289** `NppStatus nppiWarpAffineBack_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, three channels).

#### See also:

[nppiWarpAffineBack\\_8u\\_C1R](#)

**7.4.1.290** `NppStatus nppiWarpAffineBack_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, four channels).

See also:

[nppiWarpAffineBack\\_8u\\_C1R](#)

**7.4.1.291** `NppStatus nppiWarpAffineBack_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, three planes).

See also:

[nppiWarpAffineBack\\_8u\\_C1R](#)

**7.4.1.292** `NppStatus nppiWarpAffineBack_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[2][3], int interpolation)`

Inverse affine transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpAffineBack\\_8u\\_C1R](#)

**7.4.1.293** `NppStatus nppiWarpAffineQuad_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, four channels RGBA).

See also:

[nppiWarpAffineQuad\\_16u\\_C1R](#)

**7.4.1.294** `NppStatus nppiWarpAffineQuad_16u_C1R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, single channel).

This function performs affine warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpAffineQuad` uses the same formulas for pixel mapping as in `nppiWarpAffine` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but doesn't perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPPI_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPPI\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPPI\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPPI\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPPI\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPPI\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPPI\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.295** `NppStatus nppiWarpAffineQuad_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, three channels).

#### See also:

[nppiWarpAffineQuad\\_16u\\_C1R](#)

**7.4.1.296** `NppStatus nppiWarpAffineQuad_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, four channels).

See also:

[nppiWarpAffineQuad\\_16u\\_C1R](#)

**7.4.1.297** `NppStatus nppiWarpAffineQuad_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpAffineQuad\\_16u\\_C1R](#)

**7.4.1.298** `NppStatus nppiWarpAffineQuad_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpAffineQuad\\_16u\\_C1R](#)

**7.4.1.299** `NppStatus nppiWarpAffineQuad_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpAffineQuad\\_32f\\_C1R](#)

**7.4.1.300** `NppStatus nppiWarpAffineQuad_32f_C1R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, single channel).

This function performs affine warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpAffineQuad` uses the same formulas for pixel mapping as in `nppiWarpAffine` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.301** `NppStatus nppiWarpAffineQuad_32f_C3R (const Npp32f *pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f *pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, three channels).

**See also:**

[npippiWarpAffineQuad\\_32f\\_C1R](#)

**7.4.1.302** `NppStatus nppiWarpAffineQuad_32f_C4R (const Npp32f *pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f *pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, four channels).

**See also:**

[npippiWarpAffineQuad\\_32f\\_C1R](#)

**7.4.1.303** `NppStatus nppiWarpAffineQuad_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, three planes).

See also:

[nppiWarpAffineQuad\\_32f\\_C1R](#)

**7.4.1.304** `NppStatus nppiWarpAffineQuad_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit float, four planes).

See also:

[nppiWarpAffineQuad\\_32f\\_C1R](#)

**7.4.1.305** `NppStatus nppiWarpAffineQuad_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpAffineQuad\\_32s\\_C1R](#)

**7.4.1.306** `NppStatus nppiWarpAffineQuad_32s_C1R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, single channel).

This function performs affine warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpAffineQuad` uses the same formulas for pixel mapping as in `nppiWarpAffine` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

**Parameters:**

*pSrc* [Source-Image Pointer](#).  
*srcSize* Size of source image in pixels  
*nSrcStep* [Source-Image Line Step](#).  
*srcRoi* Source ROI  
*srcQuad* Source quadrangle  
*pDst* [Destination-Image Pointer](#).

*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment

**7.4.1.307** `NppStatus nppiWarpAffineQuad_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, three channels).

See also:

[nppiWarpAffineQuad\\_32s\\_C1R](#)

**7.4.1.308** `NppStatus nppiWarpAffineQuad_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, four channels).

See also:

[nppiWarpAffineQuad\\_32s\\_C1R](#)

**7.4.1.309** `NppStatus nppiWarpAffineQuad_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpAffineQuad\\_32s\\_C1R](#)

**7.4.1.310** `NppStatus nppiWarpAffineQuad_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpAffineQuad\\_32s\\_C1R](#)

**7.4.1.311** `NppStatus nppiWarpAffineQuad_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpAffineQuad\\_8u\\_C1R](#)

**7.4.1.312** `NppStatus nppiWarpAffineQuad_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, single channel).

This function performs affine warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpAffineQuad` uses the same formulas for pixel mapping as in `nppiWarpAffine` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

**Parameters:**

*pSrc* [Source-Image Pointer](#).  
*srcSize* Size of source image in pixels  
*nSrcStep* [Source-Image Line Step](#).  
*srcRoi* Source ROI  
*srcQuad* Source quadrangle  
*pDst* [Destination-Image Pointer](#).



*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI misalignment ignored, internally computed coordinates are used instead

**7.4.1.313** `NppStatus nppiWarpAffineQuad_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, three channels).

See also:

[nppiWarpAffineQuad\\_8u\\_C1R](#)

**7.4.1.314** `NppStatus nppiWarpAffineQuad_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, four channels).

See also:

[nppiWarpAffineQuad\\_8u\\_C1R](#)

**7.4.1.315** `NppStatus nppiWarpAffineQuad_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, three planes).

See also:

[nppiWarpAffineQuad\\_8u\\_C1R](#)

**7.4.1.316** `NppStatus nppiWarpAffineQuad_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Affine transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpAffineQuad\\_8u\\_C1R](#)

**7.4.1.317** `NppStatus nppiWarpPerspective_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspective\\_16u\\_C1R](#)

**7.4.1.318** `NppStatus nppiWarpPerspective_16u_C1R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = \frac{C_{00} * x + C_{01} * y + C_{02}}{C_{20} * x + C_{21} * y + C_{22}} \quad Y_{new} = \frac{C_{10} * x + C_{11} * y + C_{12}}{C_{20} * x + C_{21} * y + C_{22}}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

Parameters:

*pSrc* Source-Image Pointer.

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.319** `NppStatus nppiWarpPerspective_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, three channels).

#### See also:

[nppiWarpPerspective\\_16u\\_C1R](#)

**7.4.1.320** `NppStatus nppiWarpPerspective_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four channels).

#### See also:

[nppiWarpPerspective\\_16u\\_C1R](#)

**7.4.1.321** `NppStatus nppiWarpPerspective_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpPerspective\\_16u\\_C1R](#)

**7.4.1.322** `NppStatus nppiWarpPerspective_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpPerspective\\_16u\\_C1R](#)

**7.4.1.323** `NppStatus nppiWarpPerspective_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpPerspective\\_32f\\_C1R](#)

**7.4.1.324** `NppStatus nppiWarpPerspective_32f_C1R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = \frac{C_{00} * x + C_{01} * y + C_{02}}{C_{20} * x + C_{21} * y + C_{22}} \quad Y_{new} = \frac{C_{10} * x + C_{11} * y + C_{12}}{C_{20} * x + C_{21} * y + C_{22}}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.325** `NppStatus nppiWarpPerspective_32f_C3R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, three channels).

#### See also:

[nppiWarpPerspective\\_32f\\_C1R](#)

**7.4.1.326** `NppStatus nppiWarpPerspective_32f_C4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, four channels).

#### See also:

[nppiWarpPerspective\\_32f\\_C1R](#)

**7.4.1.327** `NppStatus nppiWarpPerspective_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, three planes).

See also:

[nppiWarpPerspective\\_32f\\_C1R](#)

**7.4.1.328** `NppStatus nppiWarpPerspective_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit float, four planes).

See also:

[nppiWarpPerspective\\_32f\\_C1R](#)

**7.4.1.329** `NppStatus nppiWarpPerspective_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpPerspective\\_32s\\_C1R](#)

**7.4.1.330** `NppStatus nppiWarpPerspective_32s_C1R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = \frac{C_{00} * x + C_{01} * y + C_{02}}{C_{20} * x + C_{21} * y + C_{22}} \quad Y_{new} = \frac{C_{10} * x + C_{11} * y + C_{12}}{C_{20} * x + C_{21} * y + C_{22}}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.331** `NppStatus nppiWarpPerspective_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, three channels).

#### See also:

[nppiWarpPerspective\\_32s\\_C1R](#)

**7.4.1.332** `NppStatus nppiWarpPerspective_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, four channels).

#### See also:

[nppiWarpPerspective\\_32s\\_C1R](#)

**7.4.1.333** `NppStatus nppiWarpPerspective_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpPerspective\\_32s\\_C1R](#)

**7.4.1.334** `NppStatus nppiWarpPerspective_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpPerspective\\_32s\\_C1R](#)

**7.4.1.335** `NppStatus nppiWarpPerspective_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspective\\_8u\\_C1R](#)

**7.4.1.336** `NppStatus nppiWarpPerspective_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$X_{new} = \frac{C_{00} * x + C_{01} * y + C_{02}}{C_{20} * x + C_{21} * y + C_{22}} \quad Y_{new} = \frac{C_{10} * x + C_{11} * y + C_{12}}{C_{20} * x + C_{21} * y + C_{22}}$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the



fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.337** `NppStatus nppiWarpPerspective_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, three channels).

#### See also:

[npplWarpPerspective\\_8u\\_C1R](#)

**7.4.1.338** `NppStatus nppiWarpPerspective_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four channels).

See also:

[nppiWarpPerspective\\_8u\\_C1R](#)

**7.4.1.339** `NppStatus nppiWarpPerspective_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, three planes).

See also:

[nppiWarpPerspective\\_8u\\_C1R](#)

**7.4.1.340** `NppStatus nppiWarpPerspective_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpPerspective\\_8u\\_C1R](#)

**7.4.1.341** `NppStatus nppiWarpPerspectiveBack_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspectiveBack\\_16u\\_C1R](#)

**7.4.1.342** `NppStatus nppiWarpPerspectiveBack_16u_C1R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpPerspectiveBack`. The function operates on source and destination regions of interest. The

perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$\frac{C_{00} * X_{new} + C_{01} * Y_{new} + C_{02}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = x \quad \frac{C_{10} * X_{new} + C_{11} * Y_{new} + C_{12}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the `srcRoi` and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if `srcRoi` has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.343** `NppStatus nppiWarpPerspectiveBack_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, three channels).

See also:

[nppiWarpPerspectiveBack\\_16u\\_C1R](#)

**7.4.1.344** `NppStatus nppiWarpPerspectiveBack_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, four channels).

See also:

[nppiWarpPerspectiveBack\\_16u\\_C1R](#)

**7.4.1.345** `NppStatus nppiWarpPerspectiveBack_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpPerspectiveBack\\_16u\\_C1R](#)

**7.4.1.346** `NppStatus nppiWarpPerspectiveBack_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpPerspectiveBack\\_16u\\_C1R](#)

**7.4.1.347** `NppStatus nppiWarpPerspectiveBack_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpPerspectiveBack\\_32f\\_C1R](#)

#### 7.4.1.348 NppStatus nppiWarpPerspectiveBack\_32f\_C1R (const Npp32f \* *pSrc*, NppiSize *srcSize*, int *nSrcStep*, NppiRect *srcRoi*, Npp32f \* *pDst*, int *nDstStep*, NppiRect *dstRoi*, const double *coeffs*[3][3], int *interpolation*)

Inverse perspective transform of an image (32bit float, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpPerspectiveBack`. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$\frac{C_{00} * X_{new} + C_{01} * Y_{new} + C_{02}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = x \quad \frac{C_{10} * X_{new} + C_{11} * Y_{new} + C_{12}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

##### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

##### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- `NPP_RECT_ERROR` Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- `NPP_WRONG_INTERSECTION_ROI_ERROR` Indicates an error condition if *srcRoi* has no intersection with the source image
- `NPP_INTERPOLATION_ERROR` Indicates an error condition if interpolation has an illegal value
- `NPP_COEFF_ERROR` Indicates an error condition if coefficient values are invalid
- `NPP_WRONG_INTERSECTION_QUAD_WARNING` Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- `NPP_MISALIGNED_DST_ROI_WARNING` Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.349** `NppStatus nppiWarpPerspectiveBack_32f_C3R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit float, three channels).

See also:

[nppiWarpPerspectiveBack\\_32f\\_C1R](#)

**7.4.1.350** `NppStatus nppiWarpPerspectiveBack_32f_C4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit float, four channels).

See also:

[nppiWarpPerspectiveBack\\_32f\\_C1R](#)

**7.4.1.351** `NppStatus nppiWarpPerspectiveBack_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit float, three planes).

See also:

[nppiWarpPerspectiveBack\\_32f\\_C1R](#)

**7.4.1.352** `NppStatus nppiWarpPerspectiveBack_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit float, four planes).

See also:

[nppiWarpPerspectiveBack\\_32f\\_C1R](#)

**7.4.1.353** `NppStatus nppiWarpPerspectiveBack_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpPerspectiveBack\\_32s\\_C1R](#)

**7.4.1.354 NppStatus nppiWarpPerspectiveBack\_32s\_C1R** (const Npp32s \* *pSrc*, NppiSize *srcSize*, int *nSrcStep*, NppiRect *srcRoi*, Npp32s \* *pDst*, int *nDstStep*, NppiRect *dstRoi*, const double *coeffs*[3][3], int *interpolation*)

Inverse perspective transform of an image (32bit signed integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpPerspectiveBack`. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$\frac{C_{00} * X_{new} + C_{01} * Y_{new} + C_{02}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = x \quad \frac{C_{10} * X_{new} + C_{11} * Y_{new} + C_{12}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be NPPI\_INTER\_NN, NPPI\_INTER\_LINEAR or NPPI\_INTER\_CUBIC

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.355** `NppStatus nppiWarpPerspectiveBack_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit signed integer, three channels).

See also:

[nppiWarpPerspectiveBack\\_32s\\_C1R](#)

**7.4.1.356** `NppStatus nppiWarpPerspectiveBack_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit signed integer, four channels).

See also:

[nppiWarpPerspectiveBack\\_32s\\_C1R](#)

**7.4.1.357** `NppStatus nppiWarpPerspectiveBack_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpPerspectiveBack\\_32s\\_C1R](#)

**7.4.1.358** `NppStatus nppiWarpPerspectiveBack_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpPerspectiveBack\\_32s\\_C1R](#)

**7.4.1.359** `NppStatus nppiWarpPerspectiveBack_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspectiveBack\\_8u\\_C1R](#)



**7.4.1.360** `NppStatus nppiWarpPerspectiveBack_8u_C1R (const Npp8u *pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u *pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, single channel).

This function operates using given transform coefficients that can be obtained by using `nppiGetPerspectiveTransform` function or set explicitly. Thus there is no need to invert coefficients in your application before calling `WarpPerspectiveBack`. The function operates on source and destination regions of interest. The perspective warp function transforms the source image pixel coordinates  $(x, y)$  according to the following formulas:

$$\frac{C_{00} * X_{new} + C_{01} * Y_{new} + C_{02}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = x \quad \frac{C_{10} * X_{new} + C_{11} * Y_{new} + C_{12}}{C_{20} * X_{new} + C_{21} * Y_{new} + C_{22}} = y$$

The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI. The functions `nppiGetPerspectiveQuad` and `nppiGetPerspectiveBound` can help with destination ROI specification.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* Source-Image Pointer.

*srcSize* Size of source image in pixels

*nSrcStep* Source-Image Line Step.

*srcRoi* Source ROI

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*coeffs* Perspective transform coefficients

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the `srcRoi` and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if `srcRoi` has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value

- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.361** `NppStatus nppiWarpPerspectiveBack_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, three channels).

See also:

[nppiWarpPerspectiveBack\\_8u\\_C1R](#)

**7.4.1.362** `NppStatus nppiWarpPerspectiveBack_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, four channels).

See also:

[nppiWarpPerspectiveBack\\_8u\\_C1R](#)

**7.4.1.363** `NppStatus nppiWarpPerspectiveBack_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, three planes).

See also:

[nppiWarpPerspectiveBack\\_8u\\_C1R](#)

**7.4.1.364** `NppStatus nppiWarpPerspectiveBack_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double coeffs[3][3], int interpolation)`

Inverse perspective transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpPerspectiveBack\\_8u\\_C1R](#)

**7.4.1.365** `NppStatus nppiWarpPerspectiveQuad_16u_AC4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspectiveQuad\\_16u\\_C1R](#)

**7.4.1.366** `NppStatus nppiWarpPerspectiveQuad_16u_C1R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, single channel).

This function performs perspective warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpPerspectiveQuad` uses the same formulas for pixel mapping as in `nppiWarpPerspective` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void*)(pDst + dstRoi.x))` and `(int)((void*)(pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPP_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the srcRoi and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if srcRoi has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.367** `NppStatus nppiWarpPerspectiveQuad_16u_C3R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, three channels).

See also:

[nppiWarpPerspectiveQuad\\_16u\\_C1R](#)

**7.4.1.368** `NppStatus nppiWarpPerspectiveQuad_16u_C4R (const Npp16u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four channels).

See also:

[nppiWarpPerspectiveQuad\\_16u\\_C1R](#)

**7.4.1.369** `NppStatus nppiWarpPerspectiveQuad_16u_P3R (const Npp16u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, three planes).

See also:

[nppiWarpPerspectiveQuad\\_16u\\_C1R](#)

**7.4.1.370** `NppStatus nppiWarpPerspectiveQuad_16u_P4R (const Npp16u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp16u * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (16bit unsigned integer, four planes).

See also:

[nppiWarpPerspectiveQuad\\_16u\\_C1R](#)

**7.4.1.371** `NppStatus nppiWarpPerspectiveQuad_32f_AC4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, four channels RGBA).

See also:

[nppiWarpPerspectiveQuad\\_32f\\_C1R](#)

**7.4.1.372** `NppStatus nppiWarpPerspectiveQuad_32f_C1R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, single channel).

This function performs perspective warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpPerspectiveQuad` uses the same formulas for pixel mapping as in `nppiWarpPerspective` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*srcSize* Size of source image in pixels

*nSrcStep* [Source-Image Line Step](#).

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the `srcRoi` and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if `srcRoi` has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.373** `NppStatus nppiWarpPerspectiveQuad_32f_C3R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, three channels).

See also:

[nppiWarpPerspectiveQuad\\_32f\\_C1R](#)

**7.4.1.374** `NppStatus nppiWarpPerspectiveQuad_32f_C4R (const Npp32f * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, four channels).

See also:

[nppiWarpPerspectiveQuad\\_32f\\_C1R](#)

**7.4.1.375** `NppStatus nppiWarpPerspectiveQuad_32f_P3R (const Npp32f * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, three planes).

See also:

[nppiWarpPerspectiveQuad\\_32f\\_C1R](#)

**7.4.1.376** `NppStatus nppiWarpPerspectiveQuad_32f_P4R (const Npp32f * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32f * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit float, four planes).

See also:

[nppiWarpPerspectiveQuad\\_32f\\_C1R](#)

**7.4.1.377** `NppStatus nppiWarpPerspectiveQuad_32s_AC4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, four channels RGBA).

See also:

[nppiWarpPerspectiveQuad\\_32s\\_C1R](#)

**7.4.1.378** `NppStatus nppiWarpPerspectiveQuad_32s_C1R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, single channel).

This function performs perspective warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpPerspectiveQuad` uses the same formulas for pixel mapping as in `nppiWarpPerspective` function. The transform coefficients are computed internally. The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

**Parameters:**

*pSrc* Source-Image Pointer.

*srcSize* Size of source image in pixels

*nSrcStep* Source-Image Line Step.

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

- [NPP\\_RECT\\_ERROR](#) Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- [NPP\\_WRONG\\_INTERSECTION\\_ROI\\_ERROR](#) Indicates an error condition if *srcRoi* has no intersection with the source image
- [NPP\\_INTERPOLATION\\_ERROR](#) Indicates an error condition if interpolation has an illegal value
- [NPP\\_COEFF\\_ERROR](#) Indicates an error condition if coefficient values are invalid
- [NPP\\_WRONG\\_INTERSECTION\\_QUAD\\_WARNING](#) Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- [NPP\\_MISALIGNED\\_DST\\_ROI\\_WARNING](#) Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.379** `NppStatus nppiWarpPerspectiveQuad_32s_C3R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, three channels).

**See also:**

[nppiWarpPerspectiveQuad\\_32s\\_C1R](#)

**7.4.1.380** `NppStatus nppiWarpPerspectiveQuad_32s_C4R (const Npp32s * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, four channels).

See also:

[nppiWarpPerspectiveQuad\\_32s\\_C1R](#)

**7.4.1.381** `NppStatus nppiWarpPerspectiveQuad_32s_P3R (const Npp32s * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, three planes).

See also:

[nppiWarpPerspectiveQuad\\_32s\\_C1R](#)

**7.4.1.382** `NppStatus nppiWarpPerspectiveQuad_32s_P4R (const Npp32s * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp32s * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (32bit signed integer, four planes).

See also:

[nppiWarpPerspectiveQuad\\_32s\\_C1R](#)

**7.4.1.383** `NppStatus nppiWarpPerspectiveQuad_8u_AC4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four channels RGBA).

See also:

[nppiWarpPerspectiveQuad\\_8u\\_C1R](#)

**7.4.1.384** `NppStatus nppiWarpPerspectiveQuad_8u_C1R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, single channel).

This function performs perspective warping of a the specified quadrangle in the source image to the specified quadrangle in the destination image. The function `nppiWarpPerspectiveQuad` uses the same formulas for pixel mapping as in `nppiWarpPerspective` function. The transform coefficients are computed internally.



The transformed part of the source image is resampled using the specified interpolation method and written to the destination ROI.

NPPI specific recommendation: The function operates using 2 types of kernels: fast and accurate. The fast method is about 4 times faster than its accurate variant, but does not perform memory access checks and requires the destination ROI to be 64 bytes aligned. Hence any destination ROI is chunked into 3 vertical stripes: the first and the third are processed by accurate kernels and the central one is processed by the fast one. In order to get the maximum available speed of execution, the projection of destination ROI onto image addresses must be 64 bytes aligned. This is always true if the values `(int)((void *) (pDst + dstRoi.x))` and `(int)((void *) (pDst + dstRoi.x + dstRoi.width))` are multiples of 64. Another rule of thumb is to specify destination ROI in such way that left and right sides of the projected image are separated from the ROI by at least 63 bytes from each side. However, this requires the whole ROI to be part of allocated memory. In case when the conditions above are not satisfied, the function may decrease in speed slightly and will return `NPPI_MISALIGNED_DST_ROI_WARNING` warning.

#### Parameters:

*pSrc* Source-Image Pointer.

*srcSize* Size of source image in pixels

*nSrcStep* Source-Image Line Step.

*srcRoi* Source ROI

*srcQuad* Source quadrangle

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*dstRoi* Destination ROI

*dstQuad* Destination quadrangle

*interpolation* Interpolation mode: can be `NPPI_INTER_NN`, `NPPI_INTER_LINEAR` or `NPPI_INTER_CUBIC`

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes

- `NPPI_RECT_ERROR` Indicates an error condition if width or height of the intersection of the *srcRoi* and source image is less than or equal to 1
- `NPPI_WRONG_INTERSECTION_ROI_ERROR` Indicates an error condition if *srcRoi* has no intersection with the source image
- `NPPI_INTERPOLATION_ERROR` Indicates an error condition if interpolation has an illegal value
- `NPPI_COEFF_ERROR` Indicates an error condition if coefficient values are invalid
- `NPPI_WRONG_INTERSECTION_QUAD_WARNING` Indicates a warning that no operation is performed if the transformed source ROI has no intersection with the destination ROI
- `NPPI_MISALIGNED_DST_ROI_WARNING` Indicates a warning that the speed of primitive execution was reduced due to destination ROI

**7.4.1.385** `NppStatus nppiWarpPerspectiveQuad_8u_C3R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, three channels).

See also:

[nppiWarpPerspectiveQuad\\_8u\\_C1R](#)

**7.4.1.386** `NppStatus nppiWarpPerspectiveQuad_8u_C4R (const Npp8u * pSrc, NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst, int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four channels).

See also:

[nppiWarpPerspectiveQuad\\_8u\\_C1R](#)

**7.4.1.387** `NppStatus nppiWarpPerspectiveQuad_8u_P3R (const Npp8u * pSrc[3], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst[3], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, three planes).

See also:

[nppiWarpPerspectiveQuad\\_8u\\_C1R](#)

**7.4.1.388** `NppStatus nppiWarpPerspectiveQuad_8u_P4R (const Npp8u * pSrc[4], NppiSize srcSize, int nSrcStep, NppiRect srcRoi, const double srcQuad[4][2], Npp8u * pDst[4], int nDstStep, NppiRect dstRoi, const double dstQuad[4][2], int interpolation)`

Perspective transform of an image (8bit unsigned integer, four planes).

See also:

[nppiWarpPerspectiveQuad\\_8u\\_C1R](#)

**7.4.1.389** `NppStatus nppiYCbCr420ToRGB_8u_P3C3R (const Npp8u * const * pSrc, int nSrcStep[3], Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr420 to packed RGB color conversion.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.390** `NppStatus nppiYCbCr420ToYCbCr411_8u_P3P2R (const Npp8u *const *pSrc, int aSrcStep[3], Npp8u *pDstY, int nDstYStep, Npp8u *pDstCbCr, int nDstCbCrStep, NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr:420 to YCbCr:411 resampling.

**Parameters:**

*pSrc* Array of pointers to the source image planes.

*aSrcStep* Array with distances in bytes between starts of consecutive lines of the source image planes.

*pDstY* [Destination-Image Pointer](#). Y-channel.

*nDstYStep* [Destination-Image Line Step](#). Y-channel.

*pDstCbCr* [Destination-Image Pointer](#). CbCr image.

*nDstCbCrStep* [Destination-Image Line Step](#). CbCr image.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.391** `NppStatus nppiYCbCr420ToYCbCr422_8u_P3R (const Npp8u *const *pSrc, int nSrcStep[3], Npp8u **pDst, int nDstStep[3], NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr:420 to YCbCr:422 resampling.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* [Destination-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

**Returns:**

[Image Data Related Error Codes](#), [ROI Related Error Codes](#)

**7.4.1.392** `NppStatus nppiYCbCr422ToRGB_8u_C2C3R (const Npp8u *pSrc, int nSrcStep, Npp8u *pDst, int nDstStep, NppiSize oSizeROI)`

2 channel 8-bit unsigned YCbCr422 to 3 channel packed RGB color conversion.  
images.

**Parameters:**

*pSrc* [Source-Image Pointer](#).

*nSrcStep* [Source-Image Line Step](#).

*pDst* [Destination-Image Pointer](#).

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.393** `NppStatus nppiYCbCr422ToYCbCr411_8u_P3R (const Npp8u *const *pSrc, int nSrcStep[3], Npp8u **pDst, int nDstStep[3], NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr:422 to YCbCr:411 resampling.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.394** `NppStatus nppiYCbCr422ToYCbCr420_8u_P3R (const Npp8u *const *pSrc, int nSrcStep[3], Npp8u **pDst, int nDstStep[3], NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr:422 to YCbCr:420 resampling.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.395** `NppStatus nppiYCbCrToRGB_8u_AC4R (const Npp8u *pSrc, int nSrcStep, Npp8u *pDst, int nDstStep, NppiSize oSizeROI)`

4 channel 8-bit unsigned packed YCbCr to RGB color conversion, not affecting Alpha.

Alpha channel is the last channel and is not processed.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.396** `NppStatus nppiYCbCrToRGB_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI)`

3 channel 8-bit unsigned packed YCbCr to RGB color conversion.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

**7.4.1.397** `NppStatus nppiYCbCrToRGB_8u_P3R (const Npp8u *const * pSrc, int nSrcStep, Npp8u ** pDst, int nDstStep, NppiSize oSizeROI)`

3 channel 8-bit unsigned planar YCbCr to RGB color conversion.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

## 7.5 NPP Signal Processing

### Memory Allocation

Signal-allocator methods for allocating 1D arrays of data in device memory.

All allocators have size parameters to specify the size of the signal (1D array) being allocated.

The allocator methods return a pointer to the newly allocated memory of appropriate type. If device-memory allocation is not possible due to resource constraints the allocators return 0 (i.e. NULL pointer).

All signal allocators allocate memory aligned such that it is beneficial to the performance of the majority of the signal-processing primitives. It is no mandatory however to use these allocators. Any valid CUDA device-memory pointers can be passed to NPP primitives.

- `Npp8u * nppsMalloc_8u` (int nSize)  
*8-bit unsigned signal allocator.*
- `Npp16u * nppsMalloc_16u` (int nSize)  
*16-bit unsigned signal allocator.*
- `Npp16s * nppsMalloc_16s` (int nSize)  
*16-bit signal allocator.*
- `Npp16sc * nppsMalloc_16sc` (int nSize)  
*16-bit complex-value signal allocator.*
- `Npp32u * nppsMalloc_32u` (int nSize)  
*32-bit unsigned signal allocator.*
- `Npp32s * nppsMalloc_32s` (int nSize)  
*32-bit integer signal allocator.*
- `Npp32sc * nppsMalloc_32sc` (int nSize)  
*32-bit complex integer signal allocator.*
- `Npp32f * nppsMalloc_32f` (int nSize)  
*32-bit float signal allocator.*
- `Npp32fc * nppsMalloc_32fc` (int nSize)  
*32-bit complex float signal allocator.*
- `Npp64s * nppsMalloc_64s` (int nSize)  
*64-bit long integer signal allocator.*
- `Npp64sc * nppsMalloc_64sc` (int nSize)  
*64-bit complex long integer signal allocator.*
- `Npp64f * nppsMalloc_64f` (int nSize)  
*64-bit float (double) signal allocator.*
- `Npp64fc * nppsMalloc_64fc` (int nSize)

*64-bit complex complex signal allocator.*

- void `nppsFree` (void \*pValues)  
*Free method for any 2D allocated memory.*

## Set

Set methods for 1D vectors of various types.

The copy methods operate on vector data given as a pointer to the underlying data-type (e.g. 8-bit vectors would be passed as pointers to `Npp8u` type) and length of the vectors, i.e. the number of items.

- `NppStatus nppsSet_8u` (`Npp8u` nValue, `Npp8u` \*pDst, int nLength)  
*8-bit unsigned char, vector set method.*
- `NppStatus nppsSet_16s` (`Npp16s` nValue, `Npp16s` \*pDst, int nLength)  
*16-bit integer, vector set method.*
- `NppStatus nppsSet_16sc` (`Npp16sc` nValue, `Npp16sc` \*pDst, int nLength)  
*16-bit integer complex, vector set method.*
- `NppStatus nppsSet_32s` (`Npp32s` nValue, `Npp32s` \*pDst, int nLength)  
*32-bit integer, vector set method.*
- `NppStatus nppsSet_32sc` (`Npp32sc` nValue, `Npp32sc` \*pDst, int nLength)  
*32-bit integer complex, vector set method.*
- `NppStatus nppsSet_32f` (`Npp32f` nValue, `Npp32f` \*pDst, int nLength)  
*32-bit float, vector set method.*
- `NppStatus nppsSet_32fc` (`Npp32fc` nValue, `Npp32fc` \*pDst, int nLength)  
*32-bit float complex, vector set method.*
- `NppStatus nppsSet_64s` (`Npp64s` nValue, `Npp64s` \*pDst, int nLength)  
*64-bit long long integer, vector set method.*
- `NppStatus nppsSet_64sc` (`Npp64sc` nValue, `Npp64sc` \*pDst, int nLength)  
*64-bit long long integer complex, vector set method.*
- `NppStatus nppsSet_64f` (`Npp64f` nValue, `Npp64f` \*pDst, int nLength)  
*64-bit double, vector set method.*
- `NppStatus nppsSet_64fc` (`Npp64fc` nValue, `Npp64fc` \*pDst, int nLength)  
*64-bit double complex, vector set method.*

## Zero

Set signals to zero.

- `NppStatus nppsZero_8u (Npp8u *pDst, int nLength)`  
*8-bit unsigned char, vector zero method.*
- `NppStatus nppsZero_16s (Npp16s *pDst, int nLength)`  
*16-bit integer, vector zero method.*
- `NppStatus nppsZero_16sc (Npp16sc *pDst, int nLength)`  
*16-bit integer complex, vector zero method.*
- `NppStatus nppsZero_32s (Npp32s *pDst, int nLength)`  
*32-bit integer, vector zero method.*
- `NppStatus nppsZero_32sc (Npp32sc *pDst, int nLength)`  
*32-bit integer complex, vector zero method.*
- `NppStatus nppsZero_32f (Npp32f *pDst, int nLength)`  
*32-bit float, vector zero method.*
- `NppStatus nppsZero_32fc (Npp32fc *pDst, int nLength)`  
*32-bit float complex, vector zero method.*
- `NppStatus nppsZero_64s (Npp64s *pDst, int nLength)`  
*64-bit long long integer, vector zero method.*
- `NppStatus nppsZero_64sc (Npp64sc *pDst, int nLength)`  
*64-bit long long integer complex, vector zero method.*
- `NppStatus nppsZero_64f (Npp64f *pDst, int nLength)`  
*64-bit double, vector zero method.*
- `NppStatus nppsZero_64fc (Npp64fc *pDst, int nLength)`  
*64-bit double complex, vector zero method.*

## Copy

Copy methods for various type signals.

Copy methods operate on signal data given as a pointer to the underlying data-type (e.g. 8-bit vectors would be passed as pointers to `Npp8u` type) and length of the vectors, i.e. the number of items.

- `NppStatus nppsCopy_8u (const Npp8u *pSrc, Npp8u *pDst, int len)`  
*8-bit unsigned char, vector copy method*
- `NppStatus nppsCopy_16s (const Npp16s *pSrc, Npp16s *pDst, int len)`  
*16-bit signed short, vector copy method.*



- `NppStatus nppsCopy_32s` (const `Npp32s` \*pSrc, `Npp32s` \*pDst, int nLength)  
*32-bit signed integer, vector copy method.*
- `NppStatus nppsCopy_32f` (const `Npp32f` \*pSrc, `Npp32f` \*pDst, int len)  
*32-bit float, vector copy method.*
- `NppStatus nppsCopy_64s` (const `Npp64s` \*pSrc, `Npp64s` \*pDst, int len)  
*64-bit signed integer, vector copy method.*
- `NppStatus nppsCopy_16sc` (const `Npp16sc` \*pSrc, `Npp16sc` \*pDst, int len)  
*16-bit complex short, vector copy method.*
- `NppStatus nppsCopy_32sc` (const `Npp32sc` \*pSrc, `Npp32sc` \*pDst, int len)  
*32-bit complex signed integer, vector copy method.*
- `NppStatus nppsCopy_32fc` (const `Npp32fc` \*pSrc, `Npp32fc` \*pDst, int len)  
*32-bit complex float, vector copy method.*
- `NppStatus nppsCopy_64sc` (const `Npp64sc` \*pSrc, `Npp64sc` \*pDst, int len)  
*64-bit complex signed integer, vector copy method.*
- `NppStatus nppsCopy_64fc` (const `Npp64fc` \*pSrc, `Npp64fc` \*pDst, int len)  
*64-bit complex double, vector copy method.*

## Statistical Functions

Functions that provide global signal statistics like: average, standard deviation, minimum, etc.

- `NppStatus nppsReductionGetBufferSize_8u` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 8u reductions.*
- `NppStatus nppsReductionGetBufferSize_16s` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 16s reductions.*
- `NppStatus nppsReductionGetBufferSize_16u` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 16u reductions.*
- `NppStatus nppsReductionGetBufferSize_16s_Sfs` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 16s reductions with integer-results scaling.*
- `NppStatus nppsReductionGetBufferSize_16sc` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 16sc reductions.*
- `NppStatus nppsReductionGetBufferSize_16sc_Sfs` (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 16sc reductions with integer-results scaling.*
- `NppStatus nppsReductionGetBufferSize_32s` (int nLength, int \*hpBufferSize)

*Device-buffer size (in bytes) for 32s reductions.*

- [NppStatus nppsReductionGetBufferSize\\_32u](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 32u reductions.*
- [NppStatus nppsReductionGetBufferSize\\_32s\\_Sfs](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 32s reductions with integer-results scaling.*
- [NppStatus nppsReductionGetBufferSize\\_32sc](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 32sc reductions.*
- [NppStatus nppsReductionGetBufferSize\\_32f](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 32f reductions.*
- [NppStatus nppsReductionGetBufferSize\\_32fc](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 32fc reductions.*
- [NppStatus nppsReductionGetBufferSize\\_64s](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 64s reductions.*
- [NppStatus nppsReductionGetBufferSize\\_64f](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 64f reductions.*
- [NppStatus nppsReductionGetBufferSize\\_64fc](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for 64fc reductions.*
- [NppStatus nppsSum\\_32f](#) (const [Npp32f](#) \*pSrc, int nLength, [Npp32f](#) \*pSum, [NppHintAlgorithm](#) eHint, [Npp8u](#) \*pDeviceBuffer)  
*32-bit float vector sum method*
- [NppStatus nppsSum\\_32fc](#) (const [Npp32fc](#) \*pSrc, int nLength, [Npp32fc](#) \*pSum, [NppHintAlgorithm](#) eHint, [Npp8u](#) \*pDeviceBuffer)  
*32-bit float complex vector sum method*
- [NppStatus nppsSum\\_64f](#) (const [Npp64f](#) \*pSrc, int nLength, [Npp64f](#) \*pSum, [Npp8u](#) \*pDeviceBuffer)  
*64-bit double vector sum method*
- [NppStatus nppsSum\\_64fc](#) (const [Npp64fc](#) \*pSrc, int nLength, [Npp64fc](#) \*pSum, [Npp8u](#) \*pDeviceBuffer)  
*64-bit double complex vector sum method*
- [NppStatus nppsSum\\_16s\\_Sfs](#) (const [Npp16s](#) \*pSrc, int nLength, [Npp16s](#) \*pSum, int nScaleFactor, [Npp8u](#) \*pDeviceBuffer)  
*16-bit short vector sum with integer scaling method*
- [NppStatus nppsSum\\_32s\\_Sfs](#) (const [Npp32s](#) \*pSrc, int nLength, [Npp32s](#) \*pSum, int nScaleFactor, [Npp8u](#) \*pDeviceBuffer)  
*32-bit integer vector sum with integer scaling method*

- **NppStatus nppsSum\_16sc\_Sfs** (const **Npp16sc** \*pSrc, int nLength, **Npp16sc** \*pSum, int nScaleFactor, **Npp8u** \*pDeviceBuffer)  
*16-bit short complex vector sum with integer scaling method*
- **NppStatus nppsSum\_16sc32sc\_Sfs** (const **Npp16sc** \*pSrc, int nLength, **Npp32sc** \*pSum, int nScaleFactor, **Npp8u** \*pDeviceBuffer)  
*16-bit short complex vector sum (32bit int complex) with integer scaling method*
- **NppStatus nppsSum\_16s32s\_Sfs** (const **Npp16s** \*pSrc, int nLength, **Npp32s** \*pSum, int nScaleFactor, **Npp8u** \*pDeviceBuffer)  
*16-bit integer vector sum (32bit) with integer scaling method*
- **NppStatus nppsMax\_16s** (const **Npp16s** \*pSrc, int nLength, **Npp16s** \*pMax, **Npp8u** \*pDeviceBuffer)  
*16-bit integer vector max method*
- **NppStatus nppsMax\_32s** (const **Npp32s** \*pSrc, int nLength, **Npp32s** \*pMax, **Npp8u** \*pDeviceBuffer)  
*32-bit integer vector max method*
- **NppStatus nppsMax\_32f** (const **Npp32f** \*pSrc, int nLength, **Npp32f** \*pMax, **Npp8u** \*pDeviceBuffer)  
*32-bit float vector max method*
- **NppStatus nppsMax\_64f** (const **Npp64f** \*pSrc, int nLength, **Npp64f** \*pMax, **Npp8u** \*pDeviceBuffer)  
*64-bit float vector max method*
- **NppStatus nppsMin\_16s** (const **Npp16s** \*pSrc, int nLength, **Npp16s** \*pMin, **Npp8u** \*pDeviceBuffer)  
*16-bit integer vector min method*
- **NppStatus nppsMin\_32s** (const **Npp32s** \*pSrc, int nLength, **Npp32s** \*pMin, **Npp8u** \*pDeviceBuffer)  
*32-bit integer vector min method*
- **NppStatus nppsMin\_32f** (const **Npp32f** \*pSrc, int nLength, **Npp32f** \*pMin, **Npp8u** \*pDeviceBuffer)  
*32-bit integer vector min method*
- **NppStatus nppsMin\_64f** (const **Npp64f** \*pSrc, int nLength, **Npp64f** \*pMin, **Npp8u** \*pDeviceBuffer)  
*64-bit integer vector min method*
- **NppStatus nppsMinMaxGetBufferSize\_8u** (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_8u.*
- **NppStatus nppsMinMaxGetBufferSize\_16s** (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_16s.*
- **NppStatus nppsMinMaxGetBufferSize\_16u** (int nLength, int \*hpBufferSize)

*Device-buffer size (in bytes) for nppsMinMax\_16u.*

- [NppStatus nppsMinMaxGetBufferSize\\_32s](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_32s.*
- [NppStatus nppsMinMaxGetBufferSize\\_32u](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_32u.*
- [NppStatus nppsMinMaxGetBufferSize\\_32f](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_32f.*
- [NppStatus nppsMinMaxGetBufferSize\\_64f](#) (int nLength, int \*hpBufferSize)  
*Device-buffer size (in bytes) for nppsMinMax\_64f.*
- [NppStatus nppsMinMax\\_8u](#) (const [Npp8u](#) \*pSrc, int nLength, [Npp8u](#) \*pMin, [Npp8u](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*8-bit char vector min and max method*
- [NppStatus nppsMinMax\\_16s](#) (const [Npp16s](#) \*pSrc, int nLength, [Npp16s](#) \*pMin, [Npp16s](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*16-bit signed short vector min and max method*
- [NppStatus nppsMinMax\\_16u](#) (const [Npp16u](#) \*pSrc, int nLength, [Npp16u](#) \*pMin, [Npp16u](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*16-bit unsigned short vector min and max method*
- [NppStatus nppsMinMax\\_32u](#) (const [Npp32u](#) \*pSrc, int nLength, [Npp32u](#) \*pMin, [Npp32u](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*32-bit unsigned int vector min and max method*
- [NppStatus nppsMinMax\\_32s](#) (const [Npp32s](#) \*pSrc, int nLength, [Npp32s](#) \*pMin, [Npp32s](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*32-bit signed int vector min and max method*
- [NppStatus nppsMinMax\\_32f](#) (const [Npp32f](#) \*pSrc, int nLength, [Npp32f](#) \*pMin, [Npp32f](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*32-bit float vector min and max method*
- [NppStatus nppsMinMax\\_64f](#) (const [Npp64f](#) \*pSrc, int nLength, [Npp64f](#) \*pMin, [Npp64f](#) \*pMax, [Npp8u](#) \*pDeviceBuffer)  
*64-bit double vector min and max method*

## 7.5.1 Function Documentation

### 7.5.1.1 NppStatus nppsCopy\_16s (const Npp16s \*pSrc, Npp16s \*pDst, int len)

16-bit signed short, vector copy method.

#### Parameters:

*pSrc* [Source Signal Pointer](#).

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.2 NppStatus nppsCopy\_16sc (const Npp16sc \* pSrc, Npp16sc \* pDst, int len)**

16-bit complex short, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.3 NppStatus nppsCopy\_32f (const Npp32f \* pSrc, Npp32f \* pDst, int len)**

32-bit float, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.4 NppStatus nppsCopy\_32fc (const Npp32fc \* pSrc, Npp32fc \* pDst, int len)**

32-bit complex float, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.5 NppStatus nppsCopy\_32s (const Npp32s \* *pSrc*, Npp32s \* *pDst*, int *nLength*)**

32-bit signed integer, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.6 NppStatus nppsCopy\_32sc (const Npp32sc \* *pSrc*, Npp32sc \* *pDst*, int *len*)**

32-bit complex signed integer, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.7 NppStatus nppsCopy\_64fc (const Npp64fc \* *pSrc*, Npp64fc \* *pDst*, int *len*)**

64-bit complex double, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.8 NppStatus nppsCopy\_64s (const Npp64s \* *pSrc*, Npp64s \* *pDst*, int *len*)**

64-bit signed integer, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.9 NppStatus nppsCopy\_64sc (const Npp64sc \* *pSrc*, Npp64sc \* *pDst*, int *len*)**

64-bit complex signed integer, vector copy method.

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.10 NppStatus nppsCopy\_8u (const Npp8u \* *pSrc*, Npp8u \* *pDst*, int *len*)**

8-bit unsigned char, vector copy method

**Parameters:**

*pSrc* Source Signal Pointer.

*pDst* Destination Signal Pointer.

*len* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.11 void nppsFree (void \* *pValues*)**

Free method for any 2D allocated memory.

This method should be used to free memory allocated with any of the nppiMalloc\_<modifier> methods.

**Parameters:**

*pValues* A pointer to memory allocated using nppiMalloc\_<modifier>.

**7.5.1.12 Npp16s\* nppsMalloc\_16s (int *nSize*)**

16-bit signal allocator.

**Parameters:**

*nSize* Number of shorts in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.13 Npp16sc\* nppsMalloc\_16sc (int nSize)**

16-bit complex-value signal allocator.

**Parameters:**

*nSize* Number of 16-bit complex numbers in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.14 Npp16u\* nppsMalloc\_16u (int nSize)**

16-bit unsigned signal allocator.

**Parameters:**

*nSize* Number of unsigned shorts in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.15 Npp32f\* nppsMalloc\_32f (int nSize)**

32-bit float signal allocator.

**Parameters:**

*nSize* Number of floats in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.16 Npp32fc\* nppsMalloc\_32fc (int nSize)**

32-bit complex float signal allocator.

**Parameters:**

*nSize* Number of complex float values in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.



**7.5.1.17 Npp32s\* nppsMalloc\_32s (int nSize)**

32-bit integer signal allocator.

**Parameters:**

*nSize* Number of ints in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.18 Npp32sc\* nppsMalloc\_32sc (int nSize)**

32-bit complex integer signal allocator.

**Parameters:**

*nSize* Number of complex integner values in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.19 Npp32u\* nppsMalloc\_32u (int nSize)**

32-bit unsigned signal allocator.

**Parameters:**

*nSize* Number of unsigned ints in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.20 Npp64f\* nppsMalloc\_64f (int nSize)**

64-bit float (double) signal allocator.

**Parameters:**

*nSize* Number of doubles in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.21 Npp64fc\* nppsMalloc\_64fc (int nSize)**

64-bit complex complex signal allocator.

**Parameters:**

*nSize* Number of complex double values in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.22 Npp64s\* nppsMalloc\_64s (int nSize)**

64-bit long integer signal allocator.

**Parameters:**

*nSize* Number of long ints in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.23 Npp64sc\* nppsMalloc\_64sc (int nSize)**

64-bit complex long integer signal allocator.

**Parameters:**

*nSize* Number of complex long int values in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.24 Npp8u\* nppsMalloc\_8u (int nSize)**

8-bit unsigned signal allocator.

**Parameters:**

*nSize* Number of unsigned chars in the new signal.

**Returns:**

A pointer to the new signal. 0 (NULL-pointer) indicates that an error occurred during allocation.

**7.5.1.25 NppStatus nppsMax\_16s (const Npp16s \* *pSrc*, int *nLength*, Npp16s \* *pMax*, Npp8u \* *pDeviceBuffer*)**

16-bit integer vector max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMax* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.26 NppStatus nppsMax\_32f (const Npp32f \* *pSrc*, int *nLength*, Npp32f \* *pMax*, Npp8u \* *pDeviceBuffer*)**

32-bit float vector max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMax* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.27 NppStatus nppsMax\_32s (const Npp32s \* *pSrc*, int *nLength*, Npp32s \* *pMax*, Npp8u \* *pDeviceBuffer*)**

32-bit integer vector max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMax* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.28 NppStatus nppsMax\_64f (const Npp64f \* *pSrc*, int *nLength*, Npp64f \* *pMax*, Npp8u \* *pDeviceBuffer*)**

64-bit float vector max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMax* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.29 NppStatus nppsMin\_16s (const Npp16s \* *pSrc*, int *nLength*, Npp16s \* *pMin*, Npp8u \* *pDeviceBuffer*)**

16-bit integer vector min method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.30 NppStatus nppsMin\_32f (const Npp32f \* *pSrc*, int *nLength*, Npp32f \* *pMin*, Npp8u \* *pDeviceBuffer*)**

32-bit integer vector min method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.31 NppStatus nppsMin\_32s (const Npp32s \* *pSrc*, int *nLength*, Npp32s \* *pMin*, Npp8u \* *pDeviceBuffer*)**

32-bit integer vector min method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.32 NppStatus nppsMin\_64f (const Npp64f \* *pSrc*, int *nLength*, Npp64f \* *pMin*, Npp8u \* *pDeviceBuffer*)**

64-bit integer vector min method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.33 NppStatus nppsMinMax\_16s (const Npp16s \* *pSrc*, int *nLength*, Npp16s \* *pMin*, Npp16s \* *pMax*, Npp8u \* *pDeviceBuffer*)**

16-bit signed short vector min and max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.34 **NppStatus nppsMinMax\_16u** (const Npp16u \* *pSrc*, int *nLength*, Npp16u \* *pMin*, Npp16u \* *pMax*, Npp8u \* *pDeviceBuffer*)

16-bit unsigned short vector min and max method

##### Parameters:

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

##### Returns:

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.35 **NppStatus nppsMinMax\_32f** (const Npp32f \* *pSrc*, int *nLength*, Npp32f \* *pMin*, Npp32f \* *pMax*, Npp8u \* *pDeviceBuffer*)

32-bit float vector min and max method

##### Parameters:

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

##### Returns:

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.36 **NppStatus nppsMinMax\_32s** (const Npp32s \* *pSrc*, int *nLength*, Npp32s \* *pMin*, Npp32s \* *pMax*, Npp8u \* *pDeviceBuffer*)

32-bit signed int vector min and max method

##### Parameters:

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

##### Returns:

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.37 NppStatus nppsMinMax\_32u (const Npp32u \* *pSrc*, int *nLength*, Npp32u \* *pMin*, Npp32u \* *pMax*, Npp8u \* *pDeviceBuffer*)**

32-bit unsigned int vector min and max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.38 NppStatus nppsMinMax\_64f (const Npp64f \* *pSrc*, int *nLength*, Npp64f \* *pMin*, Npp64f \* *pMax*, Npp8u \* *pDeviceBuffer*)**

64-bit double vector min and max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.39 NppStatus nppsMinMax\_8u (const Npp8u \* *pSrc*, int *nLength*, Npp8u \* *pMin*, Npp8u \* *pMax*, Npp8u \* *pDeviceBuffer*)**

8-bit char vector min and max method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pMin* Pointer to the min output result.

*pMax* Pointer to the max output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.40 NppStatus nppsMinMaxGetBufferSize\_16s (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_16s.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.41 NppStatus nppsMinMaxGetBufferSize\_16u (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_16u.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.42 NppStatus nppsMinMaxGetBufferSize\_32f (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_32f.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.43 NppStatus nppsMinMaxGetBufferSize\_32s (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_32s.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS



**7.5.1.44 NppStatus nppsMinMaxGetBufferSize\_32u (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_32u.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.45 NppStatus nppsMinMaxGetBufferSize\_64f (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_64f.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.46 NppStatus nppsMinMaxGetBufferSize\_8u (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for nppsMinMax\_8u.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.47 NppStatus nppsReductionGetBufferSize\_16s (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 16s reductions.

This primitive provides the correct buffer size for nppsSum\_16s, nppsMin\_16s, nppsMax\_16s.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.48 NppStatus nppsReductionGetBufferSize\_16s\_Sfs (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 16s reductions with integer-results scaling.

This primitive provides the correct buffer size for nppsSum\_16s\_Sfs.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.49 NppStatus nppsReductionGetBufferSize\_16sc (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 16sc reductions.

This primitive provides the correct buffer size for nppsSum\_16sc, nppsMin\_16sc, nppsMax\_16sc.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.50 NppStatus nppsReductionGetBufferSize\_16sc\_Sfs (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 16sc reductions with integer-results scaling.

This primitive provides the correct buffer size for nppsSum\_16sc\_Sfs.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.51 NppStatus nppsReductionGetBufferSize\_16u (int *nLength*, int \* *hpBufferSize*)**  
[inline]

Device-buffer size (in bytes) for 16u reductions.

This primitive provides the correct buffer size for nppsSum\_16u, nppsMin\_16u, nppsMax\_16u.

**Parameters:**

*nLength* [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.52 NppStatus nppsReductionGetBufferSize\_32f (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 32f reductions.

This primitive provides the correct buffer size for `nppsSum_32f`, `nppsMin_32f`, `nppsMax_32f`.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.53 NppStatus nppsReductionGetBufferSize\_32fc (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 32fc reductions.

This primitive provides the correct buffer size for `nppsSum_32fc`, `nppsMin_32fc`, `nppsMax_32fc`.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.54 NppStatus nppsReductionGetBufferSize\_32s (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 32s reductions.

This primitive provides the correct buffer size for `nppsSum_32sc`, `nppsMin_32sc`, `nppsMax_32sc`.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.55 NppStatus nppsReductionGetBufferSize\_32s\_Sfs (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 32s reductions with integer-results scaling.

This primitive provides the correct buffer size for nppsSum\_32s\_Sfs.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.56 NppStatus nppsReductionGetBufferSize\_32sc (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 32sc reductions.

This primitive provides the correct buffer size for nppsSum\_32sc, nppsMin\_32sc, nppsMax\_32sc.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.57 NppStatus nppsReductionGetBufferSize\_32u (int *nLength*, int \* *hpBufferSize*)**  
[inline]

Device-buffer size (in bytes) for 32u reductions.

This primitive provides the correct buffer size for nppsSum\_32u, nppsMin\_32u, nppsMax\_32u.

**Parameters:**

*nLength* [Signal Length](#).

*hpBufferSize* Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.58 NppStatus nppsReductionGetBufferSize\_64f (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 64f reductions.

This primitive provides the correct buffer size for nppsSum\_64f, nppsMin\_64f, nppsMax\_64f.

**Parameters:**

*nLength* [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.59 NppStatus nppsReductionGetBufferSize\_64fc (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 64fc reductions.

This primitive provides the correct buffer size for *nppsSum\_64fc*, *nppsMin\_64fc*, *nppsMax\_64fc*.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.60 NppStatus nppsReductionGetBufferSize\_64s (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 64s reductions.

This primitive provides the correct buffer size for *nppsSum\_64s*, *nppsMin\_64s*, *nppsMax\_64s*.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.61 NppStatus nppsReductionGetBufferSize\_8u (int *nLength*, int \* *hpBufferSize*)**

Device-buffer size (in bytes) for 8u reductions.

This primitive provides the correct buffer size for *nppsSum\_8u*, *nppsMin\_8u*, *nppsMax\_8u*.

**Parameters:**

***nLength*** [Signal Length](#).

***hpBufferSize*** Required buffer size. Important: *hpBufferSize* is a *host pointer*.

**Returns:**

NPP\_SUCCESS

**7.5.1.62 NppStatus nppsSet\_16s (Npp16s *nValue*, Npp16s \* *pDst*, int *nLength*)**

16-bit integer, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.63 NppStatus nppsSet\_16sc (Npp16sc *nValue*, Npp16sc \* *pDst*, int *nLength*)**

16-bit integer complex, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.64 NppStatus nppsSet\_32f (Npp32f *nValue*, Npp32f \* *pDst*, int *nLength*)**

32-bit float, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.65 NppStatus nppsSet\_32fc (Npp32fc *nValue*, Npp32fc \* *pDst*, int *nLength*)**

32-bit float complex, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.66 NppStatus nppsSet\_32s (Npp32s *nValue*, Npp32s \* *pDst*, int *nLength*)**

32-bit integer, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.67 NppStatus nppsSet\_32sc (Npp32sc *nValue*, Npp32sc \* *pDst*, int *nLength*)**

32-bit integer complex, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.68 NppStatus nppsSet\_64f (Npp64f *nValue*, Npp64f \* *pDst*, int *nLength*)**

64-bit double, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.69 NppStatus nppsSet\_64fc (Npp64fc *nValue*, Npp64fc \* *pDst*, int *nLength*)**

64-bit double complex, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.70 NppStatus nppsSet\_64s (Npp64s *nValue*, Npp64s \* *pDst*, int *nLength*)**

64-bit long long integer, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.71 NppStatus nppsSet\_64sc (Npp64sc *nValue*, Npp64sc \* *pDst*, int *nLength*)**

64-bit long long integer complex, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.72 NppStatus nppsSet\_8u (Npp8u *nValue*, Npp8u \* *pDst*, int *nLength*)**

8-bit unsigned char, vector set method.

**Parameters:**

*nValue* Value used to initialize the vector pDst.

*pDst* Destination Signal Pointer.



*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.73 NppStatus nppsSum\_16s32s\_Sfs (const Npp16s \* pSrc, int nLength, Npp32s \* pSum, int nScaleFactor, Npp8u \* pDeviceBuffer)**

16-bit integer vector sum (32bit) with integer scaling method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pSum* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

*nScaleFactor* Integer-result scale factor.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.74 NppStatus nppsSum\_16s\_Sfs (const Npp16s \* pSrc, int nLength, Npp16s \* pSum, int nScaleFactor, Npp8u \* pDeviceBuffer)**

16-bit short vector sum with integer scaling method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pSum* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

*nScaleFactor* Integer-result scale factor.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.75 NppStatus nppsSum\_16sc32sc\_Sfs (const Npp16sc \* pSrc, int nLength, Npp32sc \* pSum, int nScaleFactor, Npp8u \* pDeviceBuffer)**

16-bit short complex vector sum (32bit int complex) with integer scaling method

**Parameters:**

*pSrc* Source Signal Pointer.

*nLength* Signal Length.

*pSum* Pointer to the output result.  
*pDeviceBuffer* Pointer to the required device memory allocation.  
*nScaleFactor* Integer-result scale factor.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.76 **NppStatus nppsSum\_16sc\_Sfs** (const Npp16sc \* *pSrc*, int *nLength*, Npp16sc \* *pSum*, int *nScaleFactor*, Npp8u \* *pDeviceBuffer*)

16-bit short complex vector sum with integer scaling method

**Parameters:**

*pSrc* [Source Signal Pointer](#).  
*nLength* [Signal Length](#).  
*pSum* Pointer to the output result.  
*pDeviceBuffer* Pointer to the required device memory allocation.  
*nScaleFactor* Integer-result scale factor.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.77 **NppStatus nppsSum\_32f** (const Npp32f \* *pSrc*, int *nLength*, Npp32f \* *pSum*, NppHintAlgorithm *eHint*, Npp8u \* *pDeviceBuffer*)

32-bit float vector sum method

**Parameters:**

*pSrc* [Source Signal Pointer](#).  
*nLength* [Signal Length](#).  
*pSum* Pointer to the output result.  
*eHint* Suggests using specific code.  
*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

#### 7.5.1.78 **NppStatus nppsSum\_32fc** (const Npp32fc \* *pSrc*, int *nLength*, Npp32fc \* *pSum*, NppHintAlgorithm *eHint*, Npp8u \* *pDeviceBuffer*)

32-bit float complex vector sum method

**Parameters:**

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pSum* Pointer to the output result.

*eHint* Suggests using specific code.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.79 NppStatus nppsSum\_32s\_Sfs (const Npp32s \* pSrc, int nLength, Npp32s \* pSum, int nScaleFactor, Npp8u \* pDeviceBuffer)**

32-bit integer vector sum with integer scaling method

**Parameters:**

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pSum* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

*nScaleFactor* Integer-result scale factor.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.80 NppStatus nppsSum\_64f (const Npp64f \* pSrc, int nLength, Npp64f \* pSum, Npp8u \* pDeviceBuffer)**

64-bit double vector sum method

**Parameters:**

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pSum* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.81 NppStatus nppsSum\_64fc (const Npp64fc \* pSrc, int nLength, Npp64fc \* pSum, Npp8u \* pDeviceBuffer)**

64-bit double complex vector sum method

**Parameters:**

*pSrc* [Source Signal Pointer](#).

*nLength* [Signal Length](#).

*pSum* Pointer to the output result.

*pDeviceBuffer* Pointer to the required device memory allocation.

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.82 NppStatus nppsZero\_16s (Npp16s \* *pDst*, int *nLength*)**

16-bit integer, vector zero method.

**Parameters:**

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.83 NppStatus nppsZero\_16sc (Npp16sc \* *pDst*, int *nLength*)**

16-bit integer complex, vector zero method.

**Parameters:**

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.84 NppStatus nppsZero\_32f (Npp32f \* *pDst*, int *nLength*)**

32-bit float, vector zero method.

**Parameters:**

*pDst* [Destination Signal Pointer](#).

*nLength* [Signal Length](#).

**Returns:**

[Signal Data Related Error Codes](#), [Length Related Error Codes](#).

**7.5.1.85 NppStatus nppsZero\_32fc (Npp32fc \* *pDst*, int *nLength*)**

32-bit float complex, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.86 NppStatus nppsZero\_32s (Npp32s \* *pDst*, int *nLength*)**

32-bit integer, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.87 NppStatus nppsZero\_32sc (Npp32sc \* *pDst*, int *nLength*)**

32-bit integer complex, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.88 NppStatus nppsZero\_64f (Npp64f \* *pDst*, int *nLength*)**

64-bit double, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.89 NppStatus nppsZero\_64fc (Npp64fc \* *pDst*, int *nLength*)**

64-bit double complex, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.90 NppStatus nppsZero\_64s (Npp64s \* *pDst*, int *nLength*)**

64-bit long long integer, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.91 NppStatus nppsZero\_64sc (Npp64sc \* *pDst*, int *nLength*)**

64-bit long long integer complex, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

**7.5.1.92 NppStatus nppsZero\_8u (Npp8u \* *pDst*, int *nLength*)**

8-bit unsigned char, vector zero method.

**Parameters:**

*pDst* Destination Signal Pointer.

*nLength* Signal Length.

**Returns:**

Signal Data Related Error Codes, Length Related Error Codes.

# Chapter 8

## Data Structure Documentation

### 8.1 Npp16sc Struct Reference

Complex Number This struct represents a short complex number.

```
#include <nppdefs.h>
```

#### Data Fields

- [Npp16s re](#)  
*Real part.*
- [Npp16s im](#)  
*Imaginary part.*

#### 8.1.1 Detailed Description

Complex Number This struct represents a short complex number.

#### 8.1.2 Field Documentation

##### 8.1.2.1 Npp16s Npp16sc::im

Imaginary part.

##### 8.1.2.2 Npp16s Npp16sc::re

Real part.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

## 8.2 Npp32fc Struct Reference

Complex Number This struct represents a single floating-point complex number.

```
#include <nppdefs.h>
```

### Data Fields

- [Npp32f re](#)  
*Real part.*
- [Npp32f im](#)  
*Imaginary part.*

### 8.2.1 Detailed Description

Complex Number This struct represents a single floating-point complex number.

### 8.2.2 Field Documentation

#### 8.2.2.1 Npp32f Npp32fc::im

Imaginary part.

#### 8.2.2.2 Npp32f Npp32fc::re

Real part.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h



## 8.3 Npp32sc Struct Reference

Complex Number This struct represents a signed int complex number.

```
#include <nppdefs.h>
```

### Data Fields

- [Npp32s re](#)  
*Real part.*
- [Npp32s im](#)  
*Imaginary part.*

### 8.3.1 Detailed Description

Complex Number This struct represents a signed int complex number.

### 8.3.2 Field Documentation

#### 8.3.2.1 Npp32s Npp32sc::im

Imaginary part.

#### 8.3.2.2 Npp32s Npp32sc::re

Real part.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

## 8.4 Npp64fc Struct Reference

Complex Number This struct represents a double floating-point complex number.

```
#include <nppdefs.h>
```

### Data Fields

- [Npp64f re](#)  
*Real part.*
- [Npp64f im](#)  
*Imaginary part.*

### 8.4.1 Detailed Description

Complex Number This struct represents a double floating-point complex number.

### 8.4.2 Field Documentation

#### 8.4.2.1 Npp64f Npp64fc::im

Imaginary part.

#### 8.4.2.2 Npp64f Npp64fc::re

Real part.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

## 8.5 Npp64sc Struct Reference

Complex Number This struct represents a long long complex number.

```
#include <nppdefs.h>
```

### Data Fields

- [Npp64s re](#)  
*Real part.*
- [Npp64s im](#)  
*Imaginary part.*

### 8.5.1 Detailed Description

Complex Number This struct represents a long long complex number.

### 8.5.2 Field Documentation

#### 8.5.2.1 Npp64s Npp64sc::im

Imaginary part.

#### 8.5.2.2 Npp64s Npp64sc::re

Real part.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

## 8.6 NppiHaarBuffer Struct Reference

```
#include <nppdefs.h>
```

### Data Fields

- `int haarBufferSize`  
*size of the buffer*
- `Npp32s * haarBuffer`  
*buffer*

### 8.6.1 Field Documentation

#### 8.6.1.1 `Npp32s* NppiHaarBuffer::haarBuffer`

buffer

#### 8.6.1.2 `int NppiHaarBuffer::haarBufferSize`

size of the buffer

The documentation for this struct was generated from the following file:

- `C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h`

## 8.7 NppiHaarClassifier\_32f Struct Reference

```
#include <nppdefs.h>
```

### Data Fields

- int [numClassifiers](#)  
*number of classifiers*
- [Npp32s](#) \* [classifiers](#)  
*packed classifier data 40 bytes each*
- [size\\_t](#) [classifierStep](#)
- [NppiSize](#) [classifierSize](#)
- [Npp32s](#) \* [counterDevice](#)

### 8.7.1 Field Documentation

#### 8.7.1.1 [Npp32s](#)\* [NppiHaarClassifier\\_32f::classifiers](#)

packed classifier data 40 bytes each

#### 8.7.1.2 [NppiSize](#) [NppiHaarClassifier\\_32f::classifierSize](#)

#### 8.7.1.3 [size\\_t](#) [NppiHaarClassifier\\_32f::classifierStep](#)

#### 8.7.1.4 [Npp32s](#)\* [NppiHaarClassifier\\_32f::counterDevice](#)

#### 8.7.1.5 [int](#) [NppiHaarClassifier\\_32f::numClassifiers](#)

number of classifiers

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

## 8.8 NppiPoint Struct Reference

2D Point

```
#include <nppdefs.h>
```

### Data Fields

- `int x`  
*x-coordinate.*
- `int y`  
*y-coordinate.*

### 8.8.1 Detailed Description

2D Point

### 8.8.2 Field Documentation

#### 8.8.2.1 `int NppiPoint::x`

x-coordinate.

#### 8.8.2.2 `int NppiPoint::y`

y-coordinate.

The documentation for this struct was generated from the following file:

- `C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h`

## 8.9 NppiRect Struct Reference

2D Rectangle This struct contains position and size information of a rectangle in two space.

```
#include <nppdefs.h>
```

### Data Fields

- `int x`  
*x-coordinate of upper left corner.*
- `int y`  
*y-coordinate of upper left corner.*
- `int width`  
*Rectangle width.*
- `int height`  
*Rectangle height.*

### 8.9.1 Detailed Description

2D Rectangle This struct contains position and size information of a rectangle in two space.

The rectangle's position is usually signified by the coordinate of its upper-left corner.

### 8.9.2 Field Documentation

#### 8.9.2.1 `int NppiRect::height`

Rectangle height.

#### 8.9.2.2 `int NppiRect::width`

Rectangle width.

#### 8.9.2.3 `int NppiRect::x`

x-coordinate of upper left corner.

#### 8.9.2.4 `int NppiRect::y`

y-coordinate of upper left corner.

The documentation for this struct was generated from the following file:

- `C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h`

## 8.10 NppiSize Struct Reference

2D Size This struct typically represents the size of a rectangular region in two space.

```
#include <nppdefs.h>
```

### Data Fields

- int [width](#)  
*Rectangle width.*
- int [height](#)  
*Rectangle height.*

### 8.10.1 Detailed Description

2D Size This struct typically represents the size of a rectangular region in two space.

### 8.10.2 Field Documentation

#### 8.10.2.1 int NppiSize::height

Rectangle height.

#### 8.10.2.2 int NppiSize::width

Rectangle width.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h



## 8.11 NppLibraryVersion Struct Reference

```
#include <nppdefs.h>
```

### Data Fields

- int `major`  
*Major version number.*
- int `minor`  
*Minor version number.*
- int `build`  
*Build number. This reflects the nightly build this release was made from.*

### 8.11.1 Field Documentation

#### 8.11.1.1 int NppLibraryVersion::build

Build number. This reflects the nightly build this release was made from.

#### 8.11.1.2 int NppLibraryVersion::major

Major version number.

#### 8.11.1.3 int NppLibraryVersion::minor

Minor version number.

The documentation for this struct was generated from the following file:

- C:/Perforce/sw/rel/gpgpu/toolkit/r3.2/NPP/npp/include/nppdefs.h

# Index

- Basic NPP Data Types, [29](#)
- build
  - NppLibraryVersion, [261](#)
- classifiers
  - NppiHaarClassifier\_32f, [257](#)
- classifierSize
  - NppiHaarClassifier\_32f, [257](#)
- classifierStep
  - NppiHaarClassifier\_32f, [257](#)
- core\_npp
  - nppGetGpuComputeCapability, [19](#)
  - nppGetGpuName, [20](#)
  - nppGetGpuNumSMs, [20](#)
  - nppGetLibVersion, [20](#)
  - nppGetMaxThreadsPerBlock, [20](#)
- counterDevice
  - NppiHaarClassifier\_32f, [257](#)
- haarBuffer
  - NppiHaarBuffer, [256](#)
- haarBufferSize
  - NppiHaarBuffer, [256](#)
- height
  - NppiRect, [259](#)
  - NppiSize, [260](#)
- im
  - Npp16sc, [251](#)
  - Npp32fc, [252](#)
  - Npp32sc, [253](#)
  - Npp64fc, [254](#)
  - Npp64sc, [255](#)
- major
  - NppLibraryVersion, [261](#)
- minor
  - NppLibraryVersion, [261](#)
- NPP Core, [19](#)
- NPP Image Processing, [31](#)
- NPP Signal Processing, [218](#)
- NPP Type Definitions and Constants, [21](#)
- Npp16s
  - npp\_basic\_types, [30](#)
- Npp16sc, [251](#)
- im, [251](#)
- re, [251](#)
- Npp16u
  - npp\_basic\_types, [30](#)
- Npp32f
  - npp\_basic\_types, [30](#)
- Npp32fc, [252](#)
- im, [252](#)
- re, [252](#)
- Npp32s
  - npp\_basic\_types, [30](#)
- Npp32sc, [253](#)
- im, [253](#)
- re, [253](#)
- Npp32u
  - npp\_basic\_types, [30](#)
- Npp64f
  - npp\_basic\_types, [30](#)
- Npp64fc, [254](#)
- im, [254](#)
- re, [254](#)
- Npp64s
  - npp\_basic\_types, [30](#)
- Npp64sc, [255](#)
- im, [255](#)
- re, [255](#)
- Npp64u
  - npp\_basic\_types, [30](#)
- Npp8s
  - npp\_basic\_types, [30](#)
- Npp8u
  - npp\_basic\_types, [30](#)
- NPP\_AFFINE\_QUAD\_INCORRECT\_WARNING
  - typedefs\_npp, [28](#)
- NPP\_ALIGNMENT\_ERROR
  - typedefs\_npp, [28](#)
- NPP\_BAD\_ARG\_ERROR
  - typedefs\_npp, [27](#)
- NPP\_BOTH\_AXIS
  - typedefs\_npp, [26](#)
- NPP\_CMP\_EQ
  - typedefs\_npp, [26](#)
- NPP\_CMP\_GREATER
  - typedefs\_npp, [26](#)
- NPP\_CMP\_GREATER\_EQ

- typedefs\_npp, 26
- NPP\_CMP\_LESS
  - typedefs\_npp, 26
- NPP\_CMP\_LESS\_EQ
  - typedefs\_npp, 26
- NPP\_COEFF\_ERROR
  - typedefs\_npp, 27
- NPP\_CUDA\_1\_0
  - typedefs\_npp, 26
- NPP\_CUDA\_1\_1
  - typedefs\_npp, 26
- NPP\_CUDA\_1\_2
  - typedefs\_npp, 26
- NPP\_CUDA\_1\_3
  - typedefs\_npp, 26
- NPP\_CUDA\_2\_0
  - typedefs\_npp, 26
- NPP\_CUDA\_KERNEL\_EXECUTION\_ERROR
  - typedefs\_npp, 28
- NPP\_CUDA\_NOT\_CAPABLE
  - typedefs\_npp, 26
- NPP\_CUDA\_UNKNOWN\_VERSION
  - typedefs\_npp, 26
- NPP\_DOUBLE\_SIZE\_WARNING
  - typedefs\_npp, 28
- NPP\_ERROR
  - typedefs\_npp, 28
- NPP\_HAAR\_CLASSIFIER\_PIXEL\_MATCH\_-  
ERROR
  - typedefs\_npp, 27
- NPP\_HISTO\_NUMBER\_OF\_LEVELS\_ERROR
  - typedefs\_npp, 28
- NPP\_HORIZONTAL\_AXIS
  - typedefs\_npp, 26
- NPP\_INTERPOLATION\_ERROR
  - typedefs\_npp, 27
- NPP\_INVALID\_INPUT
  - typedefs\_npp, 28
- NPP\_LUT\_NUMBER\_OF\_LEVELS\_ERROR
  - typedefs\_npp, 27
- NPP\_MEM\_ALLOC\_ERR
  - typedefs\_npp, 28
- NPP\_MEMCPY\_ERROR
  - typedefs\_npp, 28
- NPP\_MEMFREE\_ERR
  - typedefs\_npp, 27
- NPP\_MEMSET\_ERR
  - typedefs\_npp, 27
- NPP\_MIRROR\_FLIP\_ERR
  - typedefs\_npp, 28
- NPP\_MISALIGNED\_DST\_ROI\_WARNING
  - typedefs\_npp, 28
- NPP\_NO\_ERROR
  - typedefs\_npp, 28
- NPP\_NOT\_EVEN\_STEP\_ERROR
  - typedefs\_npp, 27
- NPP\_NOT\_IMPLEMENTED\_ERROR
  - typedefs\_npp, 28
- NPP\_NOT\_SUFFICIENT\_COMPUTE\_-  
CAPABILITY
  - typedefs\_npp, 27
- NPP\_NOT\_SUPPORTED\_MODE\_ERROR
  - typedefs\_npp, 27
- NPP\_NULL\_POINTER\_ERROR
  - typedefs\_npp, 28
- NPP\_ODD\_ROI\_WARNING
  - typedefs\_npp, 28
- NPP\_POINTER\_ERROR
  - typedefs\_npp, 28
- NPP\_QUAD\_ERROR
  - typedefs\_npp, 27
- NPP\_RECT\_ERROR
  - typedefs\_npp, 27
- NPP\_RESIZE\_FACTOR\_ERROR
  - typedefs\_npp, 27
- NPP\_RESIZE\_NO\_OPERATION\_ERROR
  - typedefs\_npp, 27
- NPP\_RND\_FINANCIAL
  - typedefs\_npp, 27
- NPP\_RND\_NEAR
  - typedefs\_npp, 27
- NPP\_RND\_ZERO
  - typedefs\_npp, 27
- NPP\_ROUND\_MODE\_NOT\_SUPPORTED\_-  
ERROR
  - typedefs\_npp, 27
- NPP\_SIZE\_ERROR
  - typedefs\_npp, 28
- NPP\_STEP\_ERROR
  - typedefs\_npp, 28
- NPP\_SUCCESS
  - typedefs\_npp, 28
- NPP\_TEXTURE\_BIND\_ERROR
  - typedefs\_npp, 27
- NPP\_VERTICAL\_AXIS
  - typedefs\_npp, 26
- NPP\_WARNING
  - typedefs\_npp, 28
- NPP\_WRONG\_INTERSECTION\_QUAD\_-  
WARNING
  - typedefs\_npp, 28
- NPP\_WRONG\_INTERSECTION\_ROI\_ERROR
  - typedefs\_npp, 27
- npp\_basic\_types
  - Npp16s, 30
  - Npp16u, 30
  - Npp32f, 30
  - Npp32s, 30

- Npp32u, 30
- Npp64f, 30
- Npp64s, 30
- Npp64u, 30
- Npp8s, 30
- Npp8u, 30
- NPP\_MAX\_16S
  - typedefs\_npp, 24
- NPP\_MAX\_16U
  - typedefs\_npp, 24
- NPP\_MAX\_32S
  - typedefs\_npp, 24
- NPP\_MAX\_32U
  - typedefs\_npp, 24
- NPP\_MAX\_64S
  - typedefs\_npp, 24
- NPP\_MAX\_8S
  - typedefs\_npp, 24
- NPP\_MAX\_8U
  - typedefs\_npp, 24
- NPP\_MAXABS\_32F
  - typedefs\_npp, 25
- NPP\_MAXABS\_64F
  - typedefs\_npp, 25
- NPP\_MIN\_16S
  - typedefs\_npp, 25
- NPP\_MIN\_16U
  - typedefs\_npp, 25
- NPP\_MIN\_32S
  - typedefs\_npp, 25
- NPP\_MIN\_32U
  - typedefs\_npp, 25
- NPP\_MIN\_64S
  - typedefs\_npp, 25
- NPP\_MIN\_8S
  - typedefs\_npp, 25
- NPP\_MIN\_8U
  - typedefs\_npp, 25
- NPP\_MINABS\_32F
  - typedefs\_npp, 25
- NPP\_MINABS\_64F
  - typedefs\_npp, 25
- nppAlgHintAccurate
  - typedefs\_npp, 26
- nppAlgHintFast
  - typedefs\_npp, 26
- nppAlgHintNone
  - typedefs\_npp, 26
- NppCmpOp
  - typedefs\_npp, 26
- nppGetGpuComputeCapability
  - core\_npp, 19
- nppGetGpuName
  - core\_npp, 20

- nppGetGpuNumSMs
  - core\_npp, 20
- nppGetLibVersion
  - core\_npp, 20
- nppGetMaxThreadsPerBlock
  - core\_npp, 20
- NppGpuComputeCapability
  - typedefs\_npp, 26
- NppHintAlgorithm
  - typedefs\_npp, 26
- nppi
  - nppiAbsDiff\_32f\_C1R, 70
  - nppiAbsDiff\_32s\_C1R, 70
  - nppiAbsDiff\_8u\_AC4R, 71
  - nppiAbsDiff\_8u\_C1R, 71
  - nppiAbsDiff\_8u\_C4R, 71
  - nppiAbsDiffC\_32f\_C1R, 72
  - nppiAdd\_32f\_C1R, 72
  - nppiAdd\_32s\_C1R, 73
  - nppiAdd\_8u\_AC4RSfs, 73
  - nppiAdd\_8u\_C1RSfs, 73
  - nppiAdd\_8u\_C4RSfs, 74
  - nppiAddC\_32f\_C1R, 74
  - nppiAddC\_32fc\_C1R, 75
  - nppiCanny\_32f8u\_C1R, 75
  - nppiCannyGetBufferSize, 76
  - nppiColorTwist32f\_8u\_AC4R, 76
  - nppiColorTwist32f\_8u\_C3R, 76
  - nppiColorTwist32f\_8u\_P3R, 77
  - nppiCompare\_32f\_C1R, 77
  - nppiCompare\_8u\_AC4R, 78
  - nppiCompare\_8u\_C4R, 78
  - nppiConvert\_16s32f\_C1R, 79
  - nppiConvert\_16s32s\_C1R, 79
  - nppiConvert\_16s8u\_AC4R, 79
  - nppiConvert\_16s8u\_C1R, 80
  - nppiConvert\_16s8u\_C4R, 80
  - nppiConvert\_16u32f\_C1R, 81
  - nppiConvert\_16u32s\_C1R, 81
  - nppiConvert\_16u8u\_AC4R, 81
  - nppiConvert\_16u8u\_C1R, 82
  - nppiConvert\_16u8u\_C4R, 82
  - nppiConvert\_32f16s\_C1R, 82
  - nppiConvert\_32f16u\_C1R, 83
  - nppiConvert\_32f8u\_C1R, 83
  - nppiConvert\_8u16s\_AC4R, 84
  - nppiConvert\_8u16s\_C1R, 84
  - nppiConvert\_8u16s\_C4R, 84
  - nppiConvert\_8u16u\_AC4R, 85
  - nppiConvert\_8u16u\_C1R, 85
  - nppiConvert\_8u16u\_C4R, 85
  - nppiConvert\_8u32f\_C1R, 86
  - nppiCopy\_16s\_AC4R, 86
  - nppiCopy\_16s\_C1R, 86

- nppiCopy\_16s\_C4R, 87
- nppiCopy\_16u\_AC4R, 87
- nppiCopy\_16u\_C1R, 87
- nppiCopy\_16u\_C4R, 88
- nppiCopy\_32f\_AC4R, 88
- nppiCopy\_32f\_C1R, 88
- nppiCopy\_32f\_C4R, 89
- nppiCopy\_32s\_AC4R, 89
- nppiCopy\_32s\_C1R, 89
- nppiCopy\_32s\_C4R, 90
- nppiCopy\_8u\_AC4R, 90
- nppiCopy\_8u\_C1R, 90
- nppiCopy\_8u\_C4R, 91
- nppiCopyConstBorder\_32s\_C1R, 91
- nppiCopyConstBorder\_8u\_AC4R, 92
- nppiCopyConstBorder\_8u\_C1R, 92
- nppiCopyConstBorder\_8u\_C4R, 93
- nppiDCTQuantFwd8x8LS\_JPEG\_8u16s\_-  
C1R, 93
- nppiDCTQuantInv8x8LS\_JPEG\_16s8u\_C1R,  
94
- nppiDilate\_8u\_C1R, 94
- nppiDilate\_8u\_C4R, 95
- nppiDiv\_32f\_C1R, 95
- nppiDiv\_32s\_C1R, 95
- nppiDiv\_8u\_AC4RSfs, 96
- nppiDiv\_8u\_C1RSfs, 96
- nppiDiv\_8u\_C4RSfs, 97
- nppiDivC\_32f\_C1R, 97
- nppiDivC\_32fc\_C1R, 98
- nppiErode\_8u\_C1R, 98
- nppiErode\_8u\_C4R, 98
- nppiEvenLevelsHost\_32s, 99
- nppiExp\_32f\_C1R, 99
- nppiFilter\_8u\_C1R, 100
- nppiFilter\_8u\_C4R, 100
- nppiFilterBox\_8u\_C1R, 101
- nppiFilterBox\_8u\_C4R, 101
- nppiFilterColumn\_8u\_C1R, 102
- nppiFilterColumn\_8u\_C4R, 102
- nppiFilterMax\_8u\_C1R, 103
- nppiFilterMax\_8u\_C4R, 103
- nppiFilterMin\_8u\_C1R, 104
- nppiFilterMin\_8u\_C4R, 104
- nppiFilterRow\_8u\_C1R, 104
- nppiFilterRow\_8u\_C4R, 105
- nppiFree, 106
- nppiGetAffineBound, 106
- nppiGetAffineQuad, 106
- nppiGetAffineTransform, 107
- nppiGetPerspectiveBound, 107
- nppiGetPerspectiveQuad, 107
- nppiGetPerspectiveTransform, 108
- nppiGraphcut\_32s8u, 108
- nppiGraphcutGetSize, 109
- nppiHistogramEven\_16s\_AC4R, 109
- nppiHistogramEven\_16s\_C1R, 110
- nppiHistogramEven\_16s\_C4R, 110
- nppiHistogramEven\_16u\_AC4R, 111
- nppiHistogramEven\_16u\_C1R, 111
- nppiHistogramEven\_16u\_C4R, 112
- nppiHistogramEven\_8u\_AC4R, 112
- nppiHistogramEven\_8u\_C1R, 113
- nppiHistogramEven\_8u\_C4R, 113
- nppiHistogramEvenGetBufferSize\_16s\_-  
AC4R, 114
- nppiHistogramEvenGetBufferSize\_16s\_C1R,  
114
- nppiHistogramEvenGetBufferSize\_16s\_C4R,  
114
- nppiHistogramEvenGetBufferSize\_16u\_-  
AC4R, 115
- nppiHistogramEvenGetBufferSize\_16u\_C1R,  
115
- nppiHistogramEvenGetBufferSize\_16u\_C4R,  
115
- nppiHistogramEvenGetBufferSize\_8u\_AC4R,  
115
- nppiHistogramEvenGetBufferSize\_8u\_C1R,  
116
- nppiHistogramEvenGetBufferSize\_8u\_C4R,  
116
- nppiHistogramRange\_16s\_AC4R, 116
- nppiHistogramRange\_16s\_C1R, 117
- nppiHistogramRange\_16s\_C4R, 117
- nppiHistogramRange\_16u\_AC4R, 118
- nppiHistogramRange\_16u\_C1R, 118
- nppiHistogramRange\_16u\_C4R, 119
- nppiHistogramRange\_32f\_AC4R, 119
- nppiHistogramRange\_32f\_C1R, 119
- nppiHistogramRange\_32f\_C4R, 120
- nppiHistogramRange\_8u\_AC4R, 120
- nppiHistogramRange\_8u\_C1R, 121
- nppiHistogramRange\_8u\_C4R, 121
- nppiHistogramRangeGetBufferSize\_16s\_-  
AC4R, 122
- nppiHistogramRangeGetBufferSize\_16s\_-  
C1R, 122
- nppiHistogramRangeGetBufferSize\_16s\_-  
C4R, 122
- nppiHistogramRangeGetBufferSize\_16u\_-  
AC4R, 122
- nppiHistogramRangeGetBufferSize\_16u\_-  
C1R, 123
- nppiHistogramRangeGetBufferSize\_16u\_-  
C4R, 123
- nppiHistogramRangeGetBufferSize\_32f\_-  
AC4R, 123

- [nppiHistogramRangeGetBufferSize\\_32f\\_C1R, 124](#)
- [nppiHistogramRangeGetBufferSize\\_32f\\_C4R, 124](#)
- [nppiHistogramRangeGetBufferSize\\_8u\\_-AC4R, 124](#)
- [nppiHistogramRangeGetBufferSize\\_8u\\_C1R, 124](#)
- [nppiHistogramRangeGetBufferSize\\_8u\\_C4R, 125](#)
- [nppiLn\\_32f\\_C1R, 125](#)
- [nppiLUT\\_Linear\\_8u\\_AC4R, 125](#)
- [nppiLUT\\_Linear\\_8u\\_C1R, 126](#)
- [nppiLUT\\_Linear\\_8u\\_C3R, 126](#)
- [nppiMagnitude\\_32fc32f\\_C1R, 127](#)
- [nppiMagnitudeSqr\\_32fc32f\\_C1R, 127](#)
- [nppiMalloc\\_16s\\_C1, 128](#)
- [nppiMalloc\\_16s\\_C4, 128](#)
- [nppiMalloc\\_16u\\_C1, 128](#)
- [nppiMalloc\\_16u\\_C3, 129](#)
- [nppiMalloc\\_16u\\_C4, 129](#)
- [nppiMalloc\\_32f\\_C1, 129](#)
- [nppiMalloc\\_32f\\_C2, 129](#)
- [nppiMalloc\\_32f\\_C3, 130](#)
- [nppiMalloc\\_32f\\_C4, 130](#)
- [nppiMalloc\\_32s\\_C1, 130](#)
- [nppiMalloc\\_32s\\_C3, 131](#)
- [nppiMalloc\\_32s\\_C4, 131](#)
- [nppiMalloc\\_8u\\_C1, 131](#)
- [nppiMalloc\\_8u\\_C2, 131](#)
- [nppiMalloc\\_8u\\_C3, 132](#)
- [nppiMalloc\\_8u\\_C4, 132](#)
- [nppiMean\\_StdDev\\_8u\\_C1R, 132](#)
- [nppiMinMax\\_8u\\_C1R, 133](#)
- [nppiMinMax\\_8u\\_C4R, 133](#)
- [nppiMirror\\_8u\\_C1R, 133](#)
- [nppiMirror\\_8u\\_C4R, 134](#)
- [nppiMul\\_32f\\_C1R, 134](#)
- [nppiMul\\_32s\\_C1R, 135](#)
- [nppiMul\\_8u\\_AC4RSfs, 135](#)
- [nppiMul\\_8u\\_C1RSfs, 135](#)
- [nppiMul\\_8u\\_C4RSfs, 136](#)
- [nppiMulC\\_32f\\_C1R, 136](#)
- [nppiMulC\\_32fc\\_C1R, 137](#)
- [nppiNormDiff\\_Inf\\_8u\\_C1R, 137](#)
- [nppiNormDiff\\_L1\\_8u\\_C1R, 137](#)
- [nppiNormDiff\\_L2\\_8u\\_C1R, 138](#)
- [nppiQuantFwdRawTableInit\\_JPEG\\_8u, 138](#)
- [nppiQuantFwdTableInit\\_JPEG\\_8u16u, 138](#)
- [nppiQuantInvTableInit\\_JPEG\\_8u16u, 139](#)
- [nppiRectStdDev\\_32s32f\\_C1R, 139](#)
- [nppiReductionGetBufferHostSize\\_8u\\_C1R, 139](#)
- [nppiReductionGetBufferHostSize\\_8u\\_C4R, 140](#)
- [nppiResize\\_8u\\_C1R, 140](#)
- [nppiResize\\_8u\\_C4R, 141](#)
- [nppiRGBToYCbCr420\\_8u\\_C3P3R, 141](#)
- [nppiRGBToYCbCr422\\_8u\\_C3C2R, 142](#)
- [nppiRGBToYCbCr\\_8u\\_AC4R, 142](#)
- [nppiRGBToYCbCr\\_8u\\_C3R, 142](#)
- [nppiRGBToYCbCr\\_8u\\_P3R, 143](#)
- [nppiRotate\\_8u\\_C1R, 143](#)
- [nppiRotate\\_8u\\_C4R, 144](#)
- [nppiSet\\_16s\\_AC4MR, 144](#)
- [nppiSet\\_16s\\_AC4R, 145](#)
- [nppiSet\\_16s\\_C1MR, 145](#)
- [nppiSet\\_16s\\_C1R, 146](#)
- [nppiSet\\_16s\\_C2R, 146](#)
- [nppiSet\\_16s\\_C4CR, 146](#)
- [nppiSet\\_16s\\_C4MR, 147](#)
- [nppiSet\\_16s\\_C4R, 147](#)
- [nppiSet\\_16u\\_AC4MR, 147](#)
- [nppiSet\\_16u\\_AC4R, 148](#)
- [nppiSet\\_16u\\_C1MR, 148](#)
- [nppiSet\\_16u\\_C1R, 149](#)
- [nppiSet\\_16u\\_C2R, 149](#)
- [nppiSet\\_16u\\_C4CR, 149](#)
- [nppiSet\\_16u\\_C4MR, 150](#)
- [nppiSet\\_16u\\_C4R, 150](#)
- [nppiSet\\_32f\\_AC4MR, 150](#)
- [nppiSet\\_32f\\_AC4R, 151](#)
- [nppiSet\\_32f\\_C1MR, 151](#)
- [nppiSet\\_32f\\_C1R, 152](#)
- [nppiSet\\_32f\\_C4CR, 152](#)
- [nppiSet\\_32f\\_C4MR, 152](#)
- [nppiSet\\_32f\\_C4R, 153](#)
- [nppiSet\\_32s\\_AC4MR, 153](#)
- [nppiSet\\_32s\\_AC4R, 153](#)
- [nppiSet\\_32s\\_C1MR, 154](#)
- [nppiSet\\_32s\\_C1R, 154](#)
- [nppiSet\\_32s\\_C4CR, 154](#)
- [nppiSet\\_32s\\_C4MR, 155](#)
- [nppiSet\\_32s\\_C4R, 155](#)
- [nppiSet\\_8u\\_AC4MR, 156](#)
- [nppiSet\\_8u\\_AC4R, 156](#)
- [nppiSet\\_8u\\_C1MR, 156](#)
- [nppiSet\\_8u\\_C1R, 157](#)
- [nppiSet\\_8u\\_C4CR, 157](#)
- [nppiSet\\_8u\\_C4MR, 158](#)
- [nppiSet\\_8u\\_C4R, 158](#)
- [nppiSetDefaultQuantTable, 158](#)
- [nppiSqrIntegral\\_8u32s32f\\_C1R, 159](#)
- [nppiSub\\_32f\\_C1R, 159](#)
- [nppiSub\\_32s\\_C1R, 159](#)
- [nppiSub\\_8u\\_AC4RSfs, 160](#)
- [nppiSub\\_8u\\_C1RSfs, 160](#)

- [nppiSub\\_8u\\_C4RSfs, 161](#)
- [nppiSubC\\_32f\\_C1R, 161](#)
- [nppiSubC\\_32fc\\_C1R, 162](#)
- [nppiSum\\_8u\\_C1R, 162](#)
- [nppiSum\\_8u\\_C4R, 162](#)
- [nppiSumWindowColumn\\_8u32f\\_C1R, 163](#)
- [nppiSumWindowRow\\_8u32f\\_C1R, 163](#)
- [nppiSwapChannels\\_8u\\_C4IR, 164](#)
- [nppiThreshold\\_32f\\_C1R, 164](#)
- [nppiThreshold\\_8u\\_AC4R, 164](#)
- [nppiTranspose\\_8u\\_C1R, 165](#)
- [nppiWarpAffine\\_16u\\_AC4R, 165](#)
- [nppiWarpAffine\\_16u\\_C1R, 165](#)
- [nppiWarpAffine\\_16u\\_C3R, 167](#)
- [nppiWarpAffine\\_16u\\_C4R, 167](#)
- [nppiWarpAffine\\_16u\\_P3R, 167](#)
- [nppiWarpAffine\\_16u\\_P4R, 167](#)
- [nppiWarpAffine\\_32f\\_AC4R, 167](#)
- [nppiWarpAffine\\_32f\\_C1R, 168](#)
- [nppiWarpAffine\\_32f\\_C3R, 169](#)
- [nppiWarpAffine\\_32f\\_C4R, 169](#)
- [nppiWarpAffine\\_32f\\_P3R, 169](#)
- [nppiWarpAffine\\_32f\\_P4R, 169](#)
- [nppiWarpAffine\\_32s\\_AC4R, 169](#)
- [nppiWarpAffine\\_32s\\_C1R, 170](#)
- [nppiWarpAffine\\_32s\\_C3R, 171](#)
- [nppiWarpAffine\\_32s\\_C4R, 171](#)
- [nppiWarpAffine\\_32s\\_P3R, 171](#)
- [nppiWarpAffine\\_32s\\_P4R, 171](#)
- [nppiWarpAffine\\_8u\\_AC4R, 171](#)
- [nppiWarpAffine\\_8u\\_C1R, 172](#)
- [nppiWarpAffine\\_8u\\_C3R, 173](#)
- [nppiWarpAffine\\_8u\\_C4R, 173](#)
- [nppiWarpAffine\\_8u\\_P3R, 173](#)
- [nppiWarpAffine\\_8u\\_P4R, 173](#)
- [nppiWarpAffineBack\\_16u\\_AC4R, 174](#)
- [nppiWarpAffineBack\\_16u\\_C1R, 174](#)
- [nppiWarpAffineBack\\_16u\\_C3R, 175](#)
- [nppiWarpAffineBack\\_16u\\_C4R, 175](#)
- [nppiWarpAffineBack\\_16u\\_P3R, 175](#)
- [nppiWarpAffineBack\\_16u\\_P4R, 176](#)
- [nppiWarpAffineBack\\_32f\\_AC4R, 176](#)
- [nppiWarpAffineBack\\_32f\\_C1R, 176](#)
- [nppiWarpAffineBack\\_32f\\_C3R, 177](#)
- [nppiWarpAffineBack\\_32f\\_C4R, 177](#)
- [nppiWarpAffineBack\\_32f\\_P3R, 177](#)
- [nppiWarpAffineBack\\_32f\\_P4R, 178](#)
- [nppiWarpAffineBack\\_32s\\_AC4R, 178](#)
- [nppiWarpAffineBack\\_32s\\_C1R, 178](#)
- [nppiWarpAffineBack\\_32s\\_C3R, 179](#)
- [nppiWarpAffineBack\\_32s\\_C4R, 179](#)
- [nppiWarpAffineBack\\_32s\\_P3R, 179](#)
- [nppiWarpAffineBack\\_32s\\_P4R, 180](#)
- [nppiWarpAffineBack\\_8u\\_AC4R, 180](#)
- [nppiWarpAffineBack\\_8u\\_C1R, 180](#)
- [nppiWarpAffineBack\\_8u\\_C3R, 181](#)
- [nppiWarpAffineBack\\_8u\\_C4R, 181](#)
- [nppiWarpAffineBack\\_8u\\_P3R, 182](#)
- [nppiWarpAffineBack\\_8u\\_P4R, 182](#)
- [nppiWarpAffineQuad\\_16u\\_AC4R, 182](#)
- [nppiWarpAffineQuad\\_16u\\_C1R, 182](#)
- [nppiWarpAffineQuad\\_16u\\_C3R, 183](#)
- [nppiWarpAffineQuad\\_16u\\_C4R, 183](#)
- [nppiWarpAffineQuad\\_16u\\_P3R, 184](#)
- [nppiWarpAffineQuad\\_16u\\_P4R, 184](#)
- [nppiWarpAffineQuad\\_32f\\_AC4R, 184](#)
- [nppiWarpAffineQuad\\_32f\\_C1R, 184](#)
- [nppiWarpAffineQuad\\_32f\\_C3R, 185](#)
- [nppiWarpAffineQuad\\_32f\\_C4R, 185](#)
- [nppiWarpAffineQuad\\_32f\\_P3R, 185](#)
- [nppiWarpAffineQuad\\_32f\\_P4R, 186](#)
- [nppiWarpAffineQuad\\_32s\\_AC4R, 186](#)
- [nppiWarpAffineQuad\\_32s\\_C1R, 186](#)
- [nppiWarpAffineQuad\\_32s\\_C3R, 187](#)
- [nppiWarpAffineQuad\\_32s\\_C4R, 187](#)
- [nppiWarpAffineQuad\\_32s\\_P3R, 187](#)
- [nppiWarpAffineQuad\\_32s\\_P4R, 187](#)
- [nppiWarpAffineQuad\\_8u\\_AC4R, 188](#)
- [nppiWarpAffineQuad\\_8u\\_C1R, 188](#)
- [nppiWarpAffineQuad\\_8u\\_C3R, 189](#)
- [nppiWarpAffineQuad\\_8u\\_C4R, 189](#)
- [nppiWarpAffineQuad\\_8u\\_P3R, 189](#)
- [nppiWarpAffineQuad\\_8u\\_P4R, 189](#)
- [nppiWarpPerspective\\_16u\\_AC4R, 190](#)
- [nppiWarpPerspective\\_16u\\_C1R, 190](#)
- [nppiWarpPerspective\\_16u\\_C3R, 191](#)
- [nppiWarpPerspective\\_16u\\_C4R, 191](#)
- [nppiWarpPerspective\\_16u\\_P3R, 191](#)
- [nppiWarpPerspective\\_16u\\_P4R, 192](#)
- [nppiWarpPerspective\\_32f\\_AC4R, 192](#)
- [nppiWarpPerspective\\_32f\\_C1R, 192](#)
- [nppiWarpPerspective\\_32f\\_C3R, 193](#)
- [nppiWarpPerspective\\_32f\\_C4R, 193](#)
- [nppiWarpPerspective\\_32f\\_P3R, 193](#)
- [nppiWarpPerspective\\_32f\\_P4R, 194](#)
- [nppiWarpPerspective\\_32s\\_AC4R, 194](#)
- [nppiWarpPerspective\\_32s\\_C1R, 194](#)
- [nppiWarpPerspective\\_32s\\_C3R, 195](#)
- [nppiWarpPerspective\\_32s\\_C4R, 195](#)
- [nppiWarpPerspective\\_32s\\_P3R, 195](#)
- [nppiWarpPerspective\\_32s\\_P4R, 196](#)
- [nppiWarpPerspective\\_8u\\_AC4R, 196](#)
- [nppiWarpPerspective\\_8u\\_C1R, 196](#)
- [nppiWarpPerspective\\_8u\\_C3R, 197](#)
- [nppiWarpPerspective\\_8u\\_C4R, 197](#)
- [nppiWarpPerspective\\_8u\\_P3R, 198](#)
- [nppiWarpPerspective\\_8u\\_P4R, 198](#)
- [nppiWarpPerspectiveBack\\_16u\\_AC4R, 198](#)



- [nppiWarpPerspectiveBack\\_16u\\_C1R](#), 198
- [nppiWarpPerspectiveBack\\_16u\\_C3R](#), 199
- [nppiWarpPerspectiveBack\\_16u\\_C4R](#), 200
- [nppiWarpPerspectiveBack\\_16u\\_P3R](#), 200
- [nppiWarpPerspectiveBack\\_16u\\_P4R](#), 200
- [nppiWarpPerspectiveBack\\_32f\\_AC4R](#), 200
- [nppiWarpPerspectiveBack\\_32f\\_C1R](#), 200
- [nppiWarpPerspectiveBack\\_32f\\_C3R](#), 201
- [nppiWarpPerspectiveBack\\_32f\\_C4R](#), 202
- [nppiWarpPerspectiveBack\\_32f\\_P3R](#), 202
- [nppiWarpPerspectiveBack\\_32f\\_P4R](#), 202
- [nppiWarpPerspectiveBack\\_32s\\_AC4R](#), 202
- [nppiWarpPerspectiveBack\\_32s\\_C1R](#), 202
- [nppiWarpPerspectiveBack\\_32s\\_C3R](#), 203
- [nppiWarpPerspectiveBack\\_32s\\_C4R](#), 204
- [nppiWarpPerspectiveBack\\_32s\\_P3R](#), 204
- [nppiWarpPerspectiveBack\\_32s\\_P4R](#), 204
- [nppiWarpPerspectiveBack\\_8u\\_AC4R](#), 204
- [nppiWarpPerspectiveBack\\_8u\\_C1R](#), 204
- [nppiWarpPerspectiveBack\\_8u\\_C3R](#), 206
- [nppiWarpPerspectiveBack\\_8u\\_C4R](#), 206
- [nppiWarpPerspectiveBack\\_8u\\_P3R](#), 206
- [nppiWarpPerspectiveBack\\_8u\\_P4R](#), 206
- [nppiWarpPerspectiveQuad\\_16u\\_AC4R](#), 206
- [nppiWarpPerspectiveQuad\\_16u\\_C1R](#), 207
- [nppiWarpPerspectiveQuad\\_16u\\_C3R](#), 208
- [nppiWarpPerspectiveQuad\\_16u\\_C4R](#), 208
- [nppiWarpPerspectiveQuad\\_16u\\_P3R](#), 208
- [nppiWarpPerspectiveQuad\\_16u\\_P4R](#), 208
- [nppiWarpPerspectiveQuad\\_32f\\_AC4R](#), 208
- [nppiWarpPerspectiveQuad\\_32f\\_C1R](#), 209
- [nppiWarpPerspectiveQuad\\_32f\\_C3R](#), 209
- [nppiWarpPerspectiveQuad\\_32f\\_C4R](#), 210
- [nppiWarpPerspectiveQuad\\_32f\\_P3R](#), 210
- [nppiWarpPerspectiveQuad\\_32f\\_P4R](#), 210
- [nppiWarpPerspectiveQuad\\_32s\\_AC4R](#), 210
- [nppiWarpPerspectiveQuad\\_32s\\_C1R](#), 210
- [nppiWarpPerspectiveQuad\\_32s\\_C3R](#), 211
- [nppiWarpPerspectiveQuad\\_32s\\_C4R](#), 211
- [nppiWarpPerspectiveQuad\\_32s\\_P3R](#), 212
- [nppiWarpPerspectiveQuad\\_32s\\_P4R](#), 212
- [nppiWarpPerspectiveQuad\\_8u\\_AC4R](#), 212
- [nppiWarpPerspectiveQuad\\_8u\\_C1R](#), 212
- [nppiWarpPerspectiveQuad\\_8u\\_C3R](#), 213
- [nppiWarpPerspectiveQuad\\_8u\\_C4R](#), 214
- [nppiWarpPerspectiveQuad\\_8u\\_P3R](#), 214
- [nppiWarpPerspectiveQuad\\_8u\\_P4R](#), 214
- [nppiYCbCr420ToRGB\\_8u\\_P3C3R](#), 214
- [nppiYCbCr420ToYCbCr411\\_8u\\_P3P2R](#), 214
- [nppiYCbCr420ToYCbCr422\\_8u\\_P3R](#), 215
- [nppiYCbCr422ToRGB\\_8u\\_C2C3R](#), 215
- [nppiYCbCr422ToYCbCr411\\_8u\\_P3R](#), 216
- [nppiYCbCr422ToYCbCr420\\_8u\\_P3R](#), 216
- [nppiYCbCrToRGB\\_8u\\_AC4R](#), 216
- [nppiYCbCrToRGB\\_8u\\_C3R](#), 217
- [nppiYCbCrToRGB\\_8u\\_P3R](#), 217
- [NPPI\\_INTER\\_CUBIC](#)
  - [typedefs\\_npp](#), 27
- [NPPI\\_INTER\\_LANCZOS](#)
  - [typedefs\\_npp](#), 27
- [NPPI\\_INTER\\_LINEAR](#)
  - [typedefs\\_npp](#), 27
- [NPPI\\_INTER\\_NN](#)
  - [typedefs\\_npp](#), 27
- [NPPI\\_INTER\\_SUPER](#)
  - [typedefs\\_npp](#), 27
- [NPPI\\_SMOOTH\\_EDGE](#)
  - [typedefs\\_npp](#), 27
- [nppiAbsDiff\\_32f\\_C1R](#)
  - [nppi](#), 70
- [nppiAbsDiff\\_32s\\_C1R](#)
  - [nppi](#), 70
- [nppiAbsDiff\\_8u\\_AC4R](#)
  - [nppi](#), 71
- [nppiAbsDiff\\_8u\\_C1R](#)
  - [nppi](#), 71
- [nppiAbsDiff\\_8u\\_C4R](#)
  - [nppi](#), 71
- [nppiAbsDiffC\\_32f\\_C1R](#)
  - [nppi](#), 72
- [nppiAdd\\_32f\\_C1R](#)
  - [nppi](#), 72
- [nppiAdd\\_32s\\_C1R](#)
  - [nppi](#), 73
- [nppiAdd\\_8u\\_AC4RSfs](#)
  - [nppi](#), 73
- [nppiAdd\\_8u\\_C1RSfs](#)
  - [nppi](#), 73
- [nppiAdd\\_8u\\_C4RSfs](#)
  - [nppi](#), 74
- [nppiAddC\\_32f\\_C1R](#)
  - [nppi](#), 74
- [nppiAddC\\_32fc\\_C1R](#)
  - [nppi](#), 75
- [NppiAxis](#)
  - [typedefs\\_npp](#), 26
- [nppiCanny\\_32f8u\\_C1R](#)
  - [nppi](#), 75
- [nppiCannyGetBufferSize](#)
  - [nppi](#), 76
- [nppiColorTwist32f\\_8u\\_AC4R](#)
  - [nppi](#), 76
- [nppiColorTwist32f\\_8u\\_C3R](#)
  - [nppi](#), 76
- [nppiColorTwist32f\\_8u\\_P3R](#)
  - [nppi](#), 77
- [nppiCompare\\_32f\\_C1R](#)
  - [nppi](#), 77



- nppiCompare\_8u\_AC4R
  - nppi, 78
- nppiCompare\_8u\_C4R
  - nppi, 78
- nppiConvert\_16s32f\_C1R
  - nppi, 79
- nppiConvert\_16s32s\_C1R
  - nppi, 79
- nppiConvert\_16s8u\_AC4R
  - nppi, 79
- nppiConvert\_16s8u\_C1R
  - nppi, 80
- nppiConvert\_16s8u\_C4R
  - nppi, 80
- nppiConvert\_16u32f\_C1R
  - nppi, 81
- nppiConvert\_16u32s\_C1R
  - nppi, 81
- nppiConvert\_16u8u\_AC4R
  - nppi, 81
- nppiConvert\_16u8u\_C1R
  - nppi, 82
- nppiConvert\_16u8u\_C4R
  - nppi, 82
- nppiConvert\_32f16s\_C1R
  - nppi, 82
- nppiConvert\_32f16u\_C1R
  - nppi, 83
- nppiConvert\_32f8u\_C1R
  - nppi, 83
- nppiConvert\_8u16s\_AC4R
  - nppi, 84
- nppiConvert\_8u16s\_C1R
  - nppi, 84
- nppiConvert\_8u16s\_C4R
  - nppi, 84
- nppiConvert\_8u16u\_AC4R
  - nppi, 85
- nppiConvert\_8u16u\_C1R
  - nppi, 85
- nppiConvert\_8u16u\_C4R
  - nppi, 85
- nppiConvert\_8u32f\_C1R
  - nppi, 86
- nppiCopy\_16s\_AC4R
  - nppi, 86
- nppiCopy\_16s\_C1R
  - nppi, 86
- nppiCopy\_16s\_C4R
  - nppi, 87
- nppiCopy\_16u\_AC4R
  - nppi, 87
- nppiCopy\_16u\_C1R
  - nppi, 87
- nppiCopy\_16u\_C4R
  - nppi, 88
- nppiCopy\_32f\_AC4R
  - nppi, 88
- nppiCopy\_32f\_C1R
  - nppi, 88
- nppiCopy\_32f\_C4R
  - nppi, 89
- nppiCopy\_32s\_AC4R
  - nppi, 89
- nppiCopy\_32s\_C1R
  - nppi, 89
- nppiCopy\_32s\_C4R
  - nppi, 90
- nppiCopy\_8u\_AC4R
  - nppi, 90
- nppiCopy\_8u\_C1R
  - nppi, 90
- nppiCopy\_8u\_C4R
  - nppi, 91
- nppiCopyConstBorder\_32s\_C1R
  - nppi, 91
- nppiCopyConstBorder\_8u\_AC4R
  - nppi, 92
- nppiCopyConstBorder\_8u\_C1R
  - nppi, 92
- nppiCopyConstBorder\_8u\_C4R
  - nppi, 93
- nppiDCTQuantFwd8x8LS\_JPEG\_8u16s\_C1R
  - nppi, 93
- nppiDCTQuantInv8x8LS\_JPEG\_16s8u\_C1R
  - nppi, 94
- nppiDilate\_8u\_C1R
  - nppi, 94
- nppiDilate\_8u\_C4R
  - nppi, 95
- nppiDiv\_32f\_C1R
  - nppi, 95
- nppiDiv\_32s\_C1R
  - nppi, 95
- nppiDiv\_8u\_AC4RSfs
  - nppi, 96
- nppiDiv\_8u\_C1RSfs
  - nppi, 96
- nppiDiv\_8u\_C4RSfs
  - nppi, 97
- nppiDivC\_32f\_C1R
  - nppi, 97
- nppiDivC\_32fc\_C1R
  - nppi, 98
- nppiErode\_8u\_C1R
  - nppi, 98
- nppiErode\_8u\_C4R
  - nppi, 98

- nppiEvenLevelsHost\_32s
  - nppi, 99
- nppiExp\_32f\_C1R
  - nppi, 99
- nppiFilter\_8u\_C1R
  - nppi, 100
- nppiFilter\_8u\_C4R
  - nppi, 100
- nppiFilterBox\_8u\_C1R
  - nppi, 101
- nppiFilterBox\_8u\_C4R
  - nppi, 101
- nppiFilterColumn\_8u\_C1R
  - nppi, 102
- nppiFilterColumn\_8u\_C4R
  - nppi, 102
- nppiFilterMax\_8u\_C1R
  - nppi, 103
- nppiFilterMax\_8u\_C4R
  - nppi, 103
- nppiFilterMin\_8u\_C1R
  - nppi, 104
- nppiFilterMin\_8u\_C4R
  - nppi, 104
- nppiFilterRow\_8u\_C1R
  - nppi, 104
- nppiFilterRow\_8u\_C4R
  - nppi, 105
- nppiFree
  - nppi, 106
- nppiGetAffineBound
  - nppi, 106
- nppiGetAffineQuad
  - nppi, 106
- nppiGetAffineTransform
  - nppi, 107
- nppiGetPerspectiveBound
  - nppi, 107
- nppiGetPerspectiveQuad
  - nppi, 107
- nppiGetPerspectiveTransform
  - nppi, 108
- nppiGraphcut\_32s8u
  - nppi, 108
- nppiGraphcutGetSize
  - nppi, 109
- NppiHaarBuffer, 256
  - haarBuffer, 256
  - haarBufferSize, 256
- NppiHaarClassifier\_32f, 257
  - classifiers, 257
  - classifierSize, 257
  - classifierStep, 257
  - counterDevice, 257
  - numClassifiers, 257
- nppiHistogramEven\_16s\_AC4R
  - nppi, 109
- nppiHistogramEven\_16s\_C1R
  - nppi, 110
- nppiHistogramEven\_16s\_C4R
  - nppi, 110
- nppiHistogramEven\_16u\_AC4R
  - nppi, 111
- nppiHistogramEven\_16u\_C1R
  - nppi, 111
- nppiHistogramEven\_16u\_C4R
  - nppi, 112
- nppiHistogramEven\_8u\_AC4R
  - nppi, 112
- nppiHistogramEven\_8u\_C1R
  - nppi, 113
- nppiHistogramEven\_8u\_C4R
  - nppi, 113
- nppiHistogramEvenGetBufferSize\_16s\_AC4R
  - nppi, 114
- nppiHistogramEvenGetBufferSize\_16s\_C1R
  - nppi, 114
- nppiHistogramEvenGetBufferSize\_16s\_C4R
  - nppi, 114
- nppiHistogramEvenGetBufferSize\_16u\_AC4R
  - nppi, 115
- nppiHistogramEvenGetBufferSize\_16u\_C1R
  - nppi, 115
- nppiHistogramEvenGetBufferSize\_16u\_C4R
  - nppi, 115
- nppiHistogramEvenGetBufferSize\_8u\_AC4R
  - nppi, 115
- nppiHistogramEvenGetBufferSize\_8u\_C1R
  - nppi, 116
- nppiHistogramEvenGetBufferSize\_8u\_C4R
  - nppi, 116
- nppiHistogramRange\_16s\_AC4R
  - nppi, 116
- nppiHistogramRange\_16s\_C1R
  - nppi, 117
- nppiHistogramRange\_16s\_C4R
  - nppi, 117
- nppiHistogramRange\_16u\_AC4R
  - nppi, 118
- nppiHistogramRange\_16u\_C1R
  - nppi, 118
- nppiHistogramRange\_16u\_C4R
  - nppi, 119
- nppiHistogramRange\_32f\_AC4R
  - nppi, 119
- nppiHistogramRange\_32f\_C1R
  - nppi, 119
- nppiHistogramRange\_32f\_C4R

- nppi, 120
- nppiHistogramRange\_8u\_AC4R
  - nppi, 120
- nppiHistogramRange\_8u\_C1R
  - nppi, 121
- nppiHistogramRange\_8u\_C4R
  - nppi, 121
- nppiHistogramRangeGetBufferSize\_16s\_AC4R
  - nppi, 122
- nppiHistogramRangeGetBufferSize\_16s\_C1R
  - nppi, 122
- nppiHistogramRangeGetBufferSize\_16s\_C4R
  - nppi, 122
- nppiHistogramRangeGetBufferSize\_16u\_AC4R
  - nppi, 122
- nppiHistogramRangeGetBufferSize\_16u\_C1R
  - nppi, 123
- nppiHistogramRangeGetBufferSize\_16u\_C4R
  - nppi, 123
- nppiHistogramRangeGetBufferSize\_32f\_AC4R
  - nppi, 123
- nppiHistogramRangeGetBufferSize\_32f\_C1R
  - nppi, 124
- nppiHistogramRangeGetBufferSize\_32f\_C4R
  - nppi, 124
- nppiHistogramRangeGetBufferSize\_8u\_AC4R
  - nppi, 124
- nppiHistogramRangeGetBufferSize\_8u\_C1R
  - nppi, 124
- nppiHistogramRangeGetBufferSize\_8u\_C4R
  - nppi, 125
- NppiInterpolationMode
  - typedefs\_npp, 26
- nppiLn\_32f\_C1R
  - nppi, 125
- nppiLUT\_Linear\_8u\_AC4R
  - nppi, 125
- nppiLUT\_Linear\_8u\_C1R
  - nppi, 126
- nppiLUT\_Linear\_8u\_C3R
  - nppi, 126
- nppiMagnitude\_32fc32f\_C1R
  - nppi, 127
- nppiMagnitudeSqr\_32fc32f\_C1R
  - nppi, 127
- nppiMalloc\_16s\_C1
  - nppi, 128
- nppiMalloc\_16s\_C4
  - nppi, 128
- nppiMalloc\_16u\_C1
  - nppi, 128
- nppiMalloc\_16u\_C3
  - nppi, 129
- nppiMalloc\_16u\_C4
  - nppi, 129
- nppiMalloc\_32f\_C1
  - nppi, 129
- nppiMalloc\_32f\_C2
  - nppi, 129
- nppiMalloc\_32f\_C3
  - nppi, 130
- nppiMalloc\_32f\_C4
  - nppi, 130
- nppiMalloc\_32s\_C1
  - nppi, 130
- nppiMalloc\_32s\_C3
  - nppi, 131
- nppiMalloc\_32s\_C4
  - nppi, 131
- nppiMalloc\_8u\_C1
  - nppi, 131
- nppiMalloc\_8u\_C2
  - nppi, 131
- nppiMalloc\_8u\_C3
  - nppi, 132
- nppiMalloc\_8u\_C4
  - nppi, 132
- nppiMean\_StdDev\_8u\_C1R
  - nppi, 132
- nppiMinMax\_8u\_C1R
  - nppi, 133
- nppiMinMax\_8u\_C4R
  - nppi, 133
- nppiMirror\_8u\_C1R
  - nppi, 133
- nppiMirror\_8u\_C4R
  - nppi, 134
- nppiMul\_32f\_C1R
  - nppi, 134
- nppiMul\_32s\_C1R
  - nppi, 135
- nppiMul\_8u\_AC4RSfs
  - nppi, 135
- nppiMul\_8u\_C1RSfs
  - nppi, 135
- nppiMul\_8u\_C4RSfs
  - nppi, 136
- nppiMulC\_32f\_C1R
  - nppi, 136
- nppiMulC\_32fc\_C1R
  - nppi, 137
- nppiNormDiff\_Inf\_8u\_C1R
  - nppi, 137
- nppiNormDiff\_L1\_8u\_C1R
  - nppi, 137
- nppiNormDiff\_L2\_8u\_C1R
  - nppi, 138
- NppiPoint, 258

- x, [258](#)
- y, [258](#)
- `nppiQuantFwdRawTableInit_JPEG_8u`
  - `nppi`, [138](#)
- `nppiQuantFwdTableInit_JPEG_8u16u`
  - `nppi`, [138](#)
- `nppiQuantInvTableInit_JPEG_8u16u`
  - `nppi`, [139](#)
- `NppiRect`, [259](#)
  - height, [259](#)
  - width, [259](#)
  - x, [259](#)
  - y, [259](#)
- `nppiRectStdDev_32s32f_C1R`
  - `nppi`, [139](#)
- `nppiReductionGetBufferHostSize_8u_C1R`
  - `nppi`, [139](#)
- `nppiReductionGetBufferHostSize_8u_C4R`
  - `nppi`, [140](#)
- `nppiResize_8u_C1R`
  - `nppi`, [140](#)
- `nppiResize_8u_C4R`
  - `nppi`, [141](#)
- `nppiRGBToYCbCr420_8u_C3P3R`
  - `nppi`, [141](#)
- `nppiRGBToYCbCr422_8u_C3C2R`
  - `nppi`, [142](#)
- `nppiRGBToYCbCr_8u_AC4R`
  - `nppi`, [142](#)
- `nppiRGBToYCbCr_8u_C3R`
  - `nppi`, [142](#)
- `nppiRGBToYCbCr_8u_P3R`
  - `nppi`, [143](#)
- `nppiRotate_8u_C1R`
  - `nppi`, [143](#)
- `nppiRotate_8u_C4R`
  - `nppi`, [144](#)
- `nppiSet_16s_AC4MR`
  - `nppi`, [144](#)
- `nppiSet_16s_AC4R`
  - `nppi`, [145](#)
- `nppiSet_16s_C1MR`
  - `nppi`, [145](#)
- `nppiSet_16s_C1R`
  - `nppi`, [146](#)
- `nppiSet_16s_C2R`
  - `nppi`, [146](#)
- `nppiSet_16s_C4CR`
  - `nppi`, [146](#)
- `nppiSet_16s_C4MR`
  - `nppi`, [147](#)
- `nppiSet_16s_C4R`
  - `nppi`, [147](#)
- `nppiSet_16u_AC4MR`
  - `nppi`, [147](#)
- `nppiSet_16u_AC4R`
  - `nppi`, [148](#)
- `nppiSet_16u_C1MR`
  - `nppi`, [148](#)
- `nppiSet_16u_C1R`
  - `nppi`, [149](#)
- `nppiSet_16u_C2R`
  - `nppi`, [149](#)
- `nppiSet_16u_C4CR`
  - `nppi`, [149](#)
- `nppiSet_16u_C4MR`
  - `nppi`, [150](#)
- `nppiSet_16u_C4R`
  - `nppi`, [150](#)
- `nppiSet_32f_AC4MR`
  - `nppi`, [150](#)
- `nppiSet_32f_AC4R`
  - `nppi`, [151](#)
- `nppiSet_32f_C1MR`
  - `nppi`, [151](#)
- `nppiSet_32f_C1R`
  - `nppi`, [152](#)
- `nppiSet_32f_C4CR`
  - `nppi`, [152](#)
- `nppiSet_32f_C4MR`
  - `nppi`, [152](#)
- `nppiSet_32f_C4R`
  - `nppi`, [153](#)
- `nppiSet_32s_AC4MR`
  - `nppi`, [153](#)
- `nppiSet_32s_AC4R`
  - `nppi`, [153](#)
- `nppiSet_32s_C1MR`
  - `nppi`, [154](#)
- `nppiSet_32s_C1R`
  - `nppi`, [154](#)
- `nppiSet_32s_C4CR`
  - `nppi`, [154](#)
- `nppiSet_32s_C4MR`
  - `nppi`, [155](#)
- `nppiSet_32s_C4R`
  - `nppi`, [155](#)
- `nppiSet_8u_AC4MR`
  - `nppi`, [156](#)
- `nppiSet_8u_AC4R`
  - `nppi`, [156](#)
- `nppiSet_8u_C1MR`
  - `nppi`, [156](#)
- `nppiSet_8u_C1R`
  - `nppi`, [157](#)
- `nppiSet_8u_C4CR`
  - `nppi`, [157](#)
- `nppiSet_8u_C4MR`

- nppi, [158](#)
- nppiSet\_8u\_C4R
  - nppi, [158](#)
- nppiSetDefaultQuantTable
  - nppi, [158](#)
- NppiSize, [260](#)
  - height, [260](#)
  - width, [260](#)
- nppiSqrIntegral\_8u32s32f\_C1R
  - nppi, [159](#)
- nppiSub\_32f\_C1R
  - nppi, [159](#)
- nppiSub\_32s\_C1R
  - nppi, [159](#)
- nppiSub\_8u\_AC4RSfs
  - nppi, [160](#)
- nppiSub\_8u\_C1RSfs
  - nppi, [160](#)
- nppiSub\_8u\_C4RSfs
  - nppi, [161](#)
- nppiSubC\_32f\_C1R
  - nppi, [161](#)
- nppiSubC\_32fc\_C1R
  - nppi, [162](#)
- nppiSum\_8u\_C1R
  - nppi, [162](#)
- nppiSum\_8u\_C4R
  - nppi, [162](#)
- nppiSumWindowColumn\_8u32f\_C1R
  - nppi, [163](#)
- nppiSumWindowRow\_8u32f\_C1R
  - nppi, [163](#)
- nppiSwapChannels\_8u\_C4IR
  - nppi, [164](#)
- nppiThreshold\_32f\_C1R
  - nppi, [164](#)
- nppiThreshold\_8u\_AC4R
  - nppi, [164](#)
- nppiTranspose\_8u\_C1R
  - nppi, [165](#)
- nppiWarpAffine\_16u\_AC4R
  - nppi, [165](#)
- nppiWarpAffine\_16u\_C1R
  - nppi, [165](#)
- nppiWarpAffine\_16u\_C3R
  - nppi, [167](#)
- nppiWarpAffine\_16u\_C4R
  - nppi, [167](#)
- nppiWarpAffine\_16u\_P3R
  - nppi, [167](#)
- nppiWarpAffine\_16u\_P4R
  - nppi, [167](#)
- nppiWarpAffine\_32f\_AC4R
  - nppi, [167](#)
- nppiWarpAffine\_32f\_C1R
  - nppi, [168](#)
- nppiWarpAffine\_32f\_C3R
  - nppi, [169](#)
- nppiWarpAffine\_32f\_C4R
  - nppi, [169](#)
- nppiWarpAffine\_32f\_P3R
  - nppi, [169](#)
- nppiWarpAffine\_32f\_P4R
  - nppi, [169](#)
- nppiWarpAffine\_32s\_AC4R
  - nppi, [169](#)
- nppiWarpAffine\_32s\_C1R
  - nppi, [170](#)
- nppiWarpAffine\_32s\_C3R
  - nppi, [171](#)
- nppiWarpAffine\_32s\_C4R
  - nppi, [171](#)
- nppiWarpAffine\_32s\_P3R
  - nppi, [171](#)
- nppiWarpAffine\_32s\_P4R
  - nppi, [171](#)
- nppiWarpAffine\_8u\_AC4R
  - nppi, [171](#)
- nppiWarpAffine\_8u\_C1R
  - nppi, [172](#)
- nppiWarpAffine\_8u\_C3R
  - nppi, [173](#)
- nppiWarpAffine\_8u\_C4R
  - nppi, [173](#)
- nppiWarpAffine\_8u\_P3R
  - nppi, [173](#)
- nppiWarpAffine\_8u\_P4R
  - nppi, [173](#)
- nppiWarpAffineBack\_16u\_AC4R
  - nppi, [174](#)
- nppiWarpAffineBack\_16u\_C1R
  - nppi, [174](#)
- nppiWarpAffineBack\_16u\_C3R
  - nppi, [175](#)
- nppiWarpAffineBack\_16u\_C4R
  - nppi, [175](#)
- nppiWarpAffineBack\_16u\_P3R
  - nppi, [175](#)
- nppiWarpAffineBack\_16u\_P4R
  - nppi, [176](#)
- nppiWarpAffineBack\_32f\_AC4R
  - nppi, [176](#)
- nppiWarpAffineBack\_32f\_C1R
  - nppi, [176](#)
- nppiWarpAffineBack\_32f\_C3R
  - nppi, [177](#)
- nppiWarpAffineBack\_32f\_C4R
  - nppi, [177](#)

- nppiWarpAffineBack\_32f\_P3R  
nppi, 177
- nppiWarpAffineBack\_32f\_P4R  
nppi, 178
- nppiWarpAffineBack\_32s\_AC4R  
nppi, 178
- nppiWarpAffineBack\_32s\_C1R  
nppi, 178
- nppiWarpAffineBack\_32s\_C3R  
nppi, 179
- nppiWarpAffineBack\_32s\_C4R  
nppi, 179
- nppiWarpAffineBack\_32s\_P3R  
nppi, 179
- nppiWarpAffineBack\_32s\_P4R  
nppi, 180
- nppiWarpAffineBack\_8u\_AC4R  
nppi, 180
- nppiWarpAffineBack\_8u\_C1R  
nppi, 180
- nppiWarpAffineBack\_8u\_C3R  
nppi, 181
- nppiWarpAffineBack\_8u\_C4R  
nppi, 181
- nppiWarpAffineBack\_8u\_P3R  
nppi, 182
- nppiWarpAffineBack\_8u\_P4R  
nppi, 182
- nppiWarpAffineQuad\_16u\_AC4R  
nppi, 182
- nppiWarpAffineQuad\_16u\_C1R  
nppi, 182
- nppiWarpAffineQuad\_16u\_C3R  
nppi, 183
- nppiWarpAffineQuad\_16u\_C4R  
nppi, 183
- nppiWarpAffineQuad\_16u\_P3R  
nppi, 184
- nppiWarpAffineQuad\_16u\_P4R  
nppi, 184
- nppiWarpAffineQuad\_32f\_AC4R  
nppi, 184
- nppiWarpAffineQuad\_32f\_C1R  
nppi, 184
- nppiWarpAffineQuad\_32f\_C3R  
nppi, 185
- nppiWarpAffineQuad\_32f\_C4R  
nppi, 185
- nppiWarpAffineQuad\_32f\_P3R  
nppi, 185
- nppiWarpAffineQuad\_32f\_P4R  
nppi, 186
- nppiWarpAffineQuad\_32s\_AC4R  
nppi, 186
- nppiWarpAffineQuad\_32s\_C1R  
nppi, 186
- nppiWarpAffineQuad\_32s\_C3R  
nppi, 187
- nppiWarpAffineQuad\_32s\_C4R  
nppi, 187
- nppiWarpAffineQuad\_32s\_P3R  
nppi, 187
- nppiWarpAffineQuad\_32s\_P4R  
nppi, 187
- nppiWarpAffineQuad\_8u\_AC4R  
nppi, 188
- nppiWarpAffineQuad\_8u\_C1R  
nppi, 188
- nppiWarpAffineQuad\_8u\_C3R  
nppi, 189
- nppiWarpAffineQuad\_8u\_C4R  
nppi, 189
- nppiWarpAffineQuad\_8u\_P3R  
nppi, 189
- nppiWarpAffineQuad\_8u\_P4R  
nppi, 189
- nppiWarpPerspective\_16u\_AC4R  
nppi, 190
- nppiWarpPerspective\_16u\_C1R  
nppi, 190
- nppiWarpPerspective\_16u\_C3R  
nppi, 191
- nppiWarpPerspective\_16u\_C4R  
nppi, 191
- nppiWarpPerspective\_16u\_P3R  
nppi, 191
- nppiWarpPerspective\_16u\_P4R  
nppi, 192
- nppiWarpPerspective\_32f\_AC4R  
nppi, 192
- nppiWarpPerspective\_32f\_C1R  
nppi, 192
- nppiWarpPerspective\_32f\_C3R  
nppi, 193
- nppiWarpPerspective\_32f\_C4R  
nppi, 193
- nppiWarpPerspective\_32f\_P3R  
nppi, 193
- nppiWarpPerspective\_32f\_P4R  
nppi, 194
- nppiWarpPerspective\_32s\_AC4R  
nppi, 194
- nppiWarpPerspective\_32s\_C1R  
nppi, 194
- nppiWarpPerspective\_32s\_C3R  
nppi, 195
- nppiWarpPerspective\_32s\_C4R  
nppi, 195

- nppiWarpPerspective\_32s\_P3R
  - nppi, [195](#)
- nppiWarpPerspective\_32s\_P4R
  - nppi, [196](#)
- nppiWarpPerspective\_8u\_AC4R
  - nppi, [196](#)
- nppiWarpPerspective\_8u\_C1R
  - nppi, [196](#)
- nppiWarpPerspective\_8u\_C3R
  - nppi, [197](#)
- nppiWarpPerspective\_8u\_C4R
  - nppi, [197](#)
- nppiWarpPerspective\_8u\_P3R
  - nppi, [198](#)
- nppiWarpPerspective\_8u\_P4R
  - nppi, [198](#)
- nppiWarpPerspectiveBack\_16u\_AC4R
  - nppi, [198](#)
- nppiWarpPerspectiveBack\_16u\_C1R
  - nppi, [198](#)
- nppiWarpPerspectiveBack\_16u\_C3R
  - nppi, [199](#)
- nppiWarpPerspectiveBack\_16u\_C4R
  - nppi, [200](#)
- nppiWarpPerspectiveBack\_16u\_P3R
  - nppi, [200](#)
- nppiWarpPerspectiveBack\_16u\_P4R
  - nppi, [200](#)
- nppiWarpPerspectiveBack\_32f\_AC4R
  - nppi, [200](#)
- nppiWarpPerspectiveBack\_32f\_C1R
  - nppi, [200](#)
- nppiWarpPerspectiveBack\_32f\_C3R
  - nppi, [201](#)
- nppiWarpPerspectiveBack\_32f\_C4R
  - nppi, [202](#)
- nppiWarpPerspectiveBack\_32f\_P3R
  - nppi, [202](#)
- nppiWarpPerspectiveBack\_32f\_P4R
  - nppi, [202](#)
- nppiWarpPerspectiveBack\_32s\_AC4R
  - nppi, [202](#)
- nppiWarpPerspectiveBack\_32s\_C1R
  - nppi, [202](#)
- nppiWarpPerspectiveBack\_32s\_C3R
  - nppi, [203](#)
- nppiWarpPerspectiveBack\_32s\_C4R
  - nppi, [204](#)
- nppiWarpPerspectiveBack\_32s\_P3R
  - nppi, [204](#)
- nppiWarpPerspectiveBack\_32s\_P4R
  - nppi, [204](#)
- nppiWarpPerspectiveBack\_8u\_AC4R
  - nppi, [204](#)
- nppiWarpPerspectiveBack\_8u\_C1R
  - nppi, [204](#)
- nppiWarpPerspectiveBack\_8u\_C3R
  - nppi, [206](#)
- nppiWarpPerspectiveBack\_8u\_C4R
  - nppi, [206](#)
- nppiWarpPerspectiveBack\_8u\_P3R
  - nppi, [206](#)
- nppiWarpPerspectiveBack\_8u\_P4R
  - nppi, [206](#)
- nppiWarpPerspectiveQuad\_16u\_AC4R
  - nppi, [206](#)
- nppiWarpPerspectiveQuad\_16u\_C1R
  - nppi, [207](#)
- nppiWarpPerspectiveQuad\_16u\_C3R
  - nppi, [208](#)
- nppiWarpPerspectiveQuad\_16u\_C4R
  - nppi, [208](#)
- nppiWarpPerspectiveQuad\_16u\_P3R
  - nppi, [208](#)
- nppiWarpPerspectiveQuad\_16u\_P4R
  - nppi, [208](#)
- nppiWarpPerspectiveQuad\_32f\_AC4R
  - nppi, [208](#)
- nppiWarpPerspectiveQuad\_32f\_C1R
  - nppi, [209](#)
- nppiWarpPerspectiveQuad\_32f\_C3R
  - nppi, [209](#)
- nppiWarpPerspectiveQuad\_32f\_C4R
  - nppi, [210](#)
- nppiWarpPerspectiveQuad\_32f\_P3R
  - nppi, [210](#)
- nppiWarpPerspectiveQuad\_32f\_P4R
  - nppi, [210](#)
- nppiWarpPerspectiveQuad\_32s\_AC4R
  - nppi, [210](#)
- nppiWarpPerspectiveQuad\_32s\_C1R
  - nppi, [210](#)
- nppiWarpPerspectiveQuad\_32s\_C3R
  - nppi, [211](#)
- nppiWarpPerspectiveQuad\_32s\_C4R
  - nppi, [211](#)
- nppiWarpPerspectiveQuad\_32s\_P3R
  - nppi, [212](#)
- nppiWarpPerspectiveQuad\_32s\_P4R
  - nppi, [212](#)
- nppiWarpPerspectiveQuad\_8u\_AC4R
  - nppi, [212](#)
- nppiWarpPerspectiveQuad\_8u\_C1R
  - nppi, [212](#)
- nppiWarpPerspectiveQuad\_8u\_C3R
  - nppi, [213](#)
- nppiWarpPerspectiveQuad\_8u\_C4R
  - nppi, [214](#)

- nppiWarpPerspectiveQuad\_8u\_P3R
  - nppi, [214](#)
- nppiWarpPerspectiveQuad\_8u\_P4R
  - nppi, [214](#)
- nppiYCbCr420ToRGB\_8u\_P3C3R
  - nppi, [214](#)
- nppiYCbCr420ToYCbCr411\_8u\_P3P2R
  - nppi, [214](#)
- nppiYCbCr420ToYCbCr422\_8u\_P3R
  - nppi, [215](#)
- nppiYCbCr422ToRGB\_8u\_C2C3R
  - nppi, [215](#)
- nppiYCbCr422ToYCbCr411\_8u\_P3R
  - nppi, [216](#)
- nppiYCbCr422ToYCbCr420\_8u\_P3R
  - nppi, [216](#)
- nppiYCbCrToRGB\_8u\_AC4R
  - nppi, [216](#)
- nppiYCbCrToRGB\_8u\_C3R
  - nppi, [217](#)
- nppiYCbCrToRGB\_8u\_P3R
  - nppi, [217](#)
- NppLibraryVersion, [261](#)
  - build, [261](#)
  - major, [261](#)
  - minor, [261](#)
- NppRoundMode
  - typedefs\_npp, [27](#)
- npps
  - nppsCopy\_16s, [224](#)
  - nppsCopy\_16sc, [225](#)
  - nppsCopy\_32f, [225](#)
  - nppsCopy\_32fc, [225](#)
  - nppsCopy\_32s, [225](#)
  - nppsCopy\_32sc, [226](#)
  - nppsCopy\_64fc, [226](#)
  - nppsCopy\_64s, [226](#)
  - nppsCopy\_64sc, [227](#)
  - nppsCopy\_8u, [227](#)
  - nppsFree, [227](#)
  - nppsMalloc\_16s, [227](#)
  - nppsMalloc\_16sc, [228](#)
  - nppsMalloc\_16u, [228](#)
  - nppsMalloc\_32f, [228](#)
  - nppsMalloc\_32fc, [228](#)
  - nppsMalloc\_32s, [228](#)
  - nppsMalloc\_32sc, [229](#)
  - nppsMalloc\_32u, [229](#)
  - nppsMalloc\_64f, [229](#)
  - nppsMalloc\_64fc, [229](#)
  - nppsMalloc\_64s, [230](#)
  - nppsMalloc\_64sc, [230](#)
  - nppsMalloc\_8u, [230](#)
  - nppsMax\_16s, [230](#)
  - nppsMax\_32f, [231](#)
  - nppsMax\_32s, [231](#)
  - nppsMax\_64f, [231](#)
  - nppsMin\_16s, [232](#)
  - nppsMin\_32f, [232](#)
  - nppsMin\_32s, [232](#)
  - nppsMin\_64f, [233](#)
  - nppsMinMax\_16s, [233](#)
  - nppsMinMax\_16u, [233](#)
  - nppsMinMax\_32f, [234](#)
  - nppsMinMax\_32s, [234](#)
  - nppsMinMax\_32u, [234](#)
  - nppsMinMax\_64f, [235](#)
  - nppsMinMax\_8u, [235](#)
  - nppsMinMaxGetBufferSize\_16s, [235](#)
  - nppsMinMaxGetBufferSize\_16u, [236](#)
  - nppsMinMaxGetBufferSize\_32f, [236](#)
  - nppsMinMaxGetBufferSize\_32s, [236](#)
  - nppsMinMaxGetBufferSize\_32u, [236](#)
  - nppsMinMaxGetBufferSize\_64f, [237](#)
  - nppsMinMaxGetBufferSize\_8u, [237](#)
  - nppsReductionGetBufferSize\_16s, [237](#)
  - nppsReductionGetBufferSize\_16s\_Sfs, [237](#)
  - nppsReductionGetBufferSize\_16sc, [238](#)
  - nppsReductionGetBufferSize\_16sc\_Sfs, [238](#)
  - nppsReductionGetBufferSize\_16u, [238](#)
  - nppsReductionGetBufferSize\_32f, [239](#)
  - nppsReductionGetBufferSize\_32fc, [239](#)
  - nppsReductionGetBufferSize\_32s, [239](#)
  - nppsReductionGetBufferSize\_32s\_Sfs, [239](#)
  - nppsReductionGetBufferSize\_32sc, [240](#)
  - nppsReductionGetBufferSize\_32u, [240](#)
  - nppsReductionGetBufferSize\_64f, [240](#)
  - nppsReductionGetBufferSize\_64fc, [241](#)
  - nppsReductionGetBufferSize\_64s, [241](#)
  - nppsReductionGetBufferSize\_8u, [241](#)
  - nppsSet\_16s, [241](#)
  - nppsSet\_16sc, [242](#)
  - nppsSet\_32f, [242](#)
  - nppsSet\_32fc, [242](#)
  - nppsSet\_32s, [243](#)
  - nppsSet\_32sc, [243](#)
  - nppsSet\_64f, [243](#)
  - nppsSet\_64fc, [243](#)
  - nppsSet\_64s, [244](#)
  - nppsSet\_64sc, [244](#)
  - nppsSet\_8u, [244](#)
  - nppsSum\_16s32s\_Sfs, [245](#)
  - nppsSum\_16s\_Sfs, [245](#)
  - nppsSum\_16sc32sc\_Sfs, [245](#)
  - nppsSum\_16sc\_Sfs, [246](#)
  - nppsSum\_32f, [246](#)
  - nppsSum\_32fc, [246](#)
  - nppsSum\_32s\_Sfs, [247](#)



- nppsSum\_64f, [247](#)
- nppsSum\_64fc, [247](#)
- nppsZero\_16s, [248](#)
- nppsZero\_16sc, [248](#)
- nppsZero\_32f, [248](#)
- nppsZero\_32fc, [248](#)
- nppsZero\_32s, [249](#)
- nppsZero\_32sc, [249](#)
- nppsZero\_64f, [249](#)
- nppsZero\_64fc, [249](#)
- nppsZero\_64s, [250](#)
- nppsZero\_64sc, [250](#)
- nppsZero\_8u, [250](#)
- nppsCopy\_16s
  - npps, [224](#)
- nppsCopy\_16sc
  - npps, [225](#)
- nppsCopy\_32f
  - npps, [225](#)
- nppsCopy\_32fc
  - npps, [225](#)
- nppsCopy\_32s
  - npps, [225](#)
- nppsCopy\_32sc
  - npps, [226](#)
- nppsCopy\_64fc
  - npps, [226](#)
- nppsCopy\_64s
  - npps, [226](#)
- nppsCopy\_64sc
  - npps, [227](#)
- nppsCopy\_8u
  - npps, [227](#)
- nppsFree
  - npps, [227](#)
- nppsMalloc\_16s
  - npps, [227](#)
- nppsMalloc\_16sc
  - npps, [228](#)
- nppsMalloc\_16u
  - npps, [228](#)
- nppsMalloc\_32f
  - npps, [228](#)
- nppsMalloc\_32fc
  - npps, [228](#)
- nppsMalloc\_32s
  - npps, [228](#)
- nppsMalloc\_32sc
  - npps, [229](#)
- nppsMalloc\_32u
  - npps, [229](#)
- nppsMalloc\_64f
  - npps, [229](#)
- nppsMalloc\_64fc
  - npps, [229](#)
- npps, [229](#)
- nppsMalloc\_64s
  - npps, [230](#)
- nppsMalloc\_64sc
  - npps, [230](#)
- nppsMalloc\_8u
  - npps, [230](#)
- nppsMax\_16s
  - npps, [230](#)
- nppsMax\_32f
  - npps, [231](#)
- nppsMax\_32s
  - npps, [231](#)
- nppsMax\_64f
  - npps, [231](#)
- nppsMin\_16s
  - npps, [232](#)
- nppsMin\_32f
  - npps, [232](#)
- nppsMin\_32s
  - npps, [232](#)
- nppsMin\_64f
  - npps, [233](#)
- nppsMinMax\_16s
  - npps, [233](#)
- nppsMinMax\_16u
  - npps, [233](#)
- nppsMinMax\_32f
  - npps, [234](#)
- nppsMinMax\_32s
  - npps, [234](#)
- nppsMinMax\_32u
  - npps, [234](#)
- nppsMinMax\_64f
  - npps, [235](#)
- nppsMinMax\_8u
  - npps, [235](#)
- nppsMinMaxGetBufferSize\_16s
  - npps, [235](#)
- nppsMinMaxGetBufferSize\_16u
  - npps, [236](#)
- nppsMinMaxGetBufferSize\_32f
  - npps, [236](#)
- nppsMinMaxGetBufferSize\_32s
  - npps, [236](#)
- nppsMinMaxGetBufferSize\_32u
  - npps, [236](#)
- nppsMinMaxGetBufferSize\_64f
  - npps, [237](#)
- nppsMinMaxGetBufferSize\_8u
  - npps, [237](#)
- nppsReductionGetBufferSize\_16s
  - npps, [237](#)
- nppsReductionGetBufferSize\_16s\_Sfs

- npps, 237
- nppsReductionGetBufferSize\_16sc
  - npps, 238
- nppsReductionGetBufferSize\_16sc\_Sfs
  - npps, 238
- nppsReductionGetBufferSize\_16u
  - npps, 238
- nppsReductionGetBufferSize\_32f
  - npps, 239
- nppsReductionGetBufferSize\_32fc
  - npps, 239
- nppsReductionGetBufferSize\_32s
  - npps, 239
- nppsReductionGetBufferSize\_32s\_Sfs
  - npps, 239
- nppsReductionGetBufferSize\_32sc
  - npps, 240
- nppsReductionGetBufferSize\_32u
  - npps, 240
- nppsReductionGetBufferSize\_64f
  - npps, 240
- nppsReductionGetBufferSize\_64fc
  - npps, 241
- nppsReductionGetBufferSize\_64s
  - npps, 241
- nppsReductionGetBufferSize\_8u
  - npps, 241
- nppsSet\_16s
  - npps, 241
- nppsSet\_16sc
  - npps, 242
- nppsSet\_32f
  - npps, 242
- nppsSet\_32fc
  - npps, 242
- nppsSet\_32s
  - npps, 243
- nppsSet\_32sc
  - npps, 243
- nppsSet\_64f
  - npps, 243
- nppsSet\_64fc
  - npps, 243
- nppsSet\_64s
  - npps, 244
- nppsSet\_64sc
  - npps, 244
- nppsSet\_8u
  - npps, 244
- nppsSum\_16s32s\_Sfs
  - npps, 245
- nppsSum\_16s\_Sfs
  - npps, 245
- nppsSum\_16sc\_Sfs
  - npps, 246
- nppsSum\_32f
  - npps, 246
- nppsSum\_32fc
  - npps, 246
- nppsSum\_32s\_Sfs
  - npps, 247
- nppsSum\_64f
  - npps, 247
- nppsSum\_64fc
  - npps, 247
- NppStatus
  - typedefs\_npp, 27
- nppsZero\_16s
  - npps, 248
- nppsZero\_16sc
  - npps, 248
- nppsZero\_32f
  - npps, 248
- nppsZero\_32fc
  - npps, 248
- nppsZero\_32s
  - npps, 249
- nppsZero\_32sc
  - npps, 249
- nppsZero\_64f
  - npps, 249
- nppsZero\_64fc
  - npps, 249
- nppsZero\_64s
  - npps, 250
- nppsZero\_64sc
  - npps, 250
- nppsZero\_8u
  - npps, 250
- numClassifiers
  - NppiHaarClassifier\_32f, 257
- re
  - Npp16sc, 251
  - Npp32fc, 252
  - Npp32sc, 253
  - Npp64fc, 254
  - Npp64sc, 255
- typedefs\_npp
  - NPP\_AFFINE\_QUAD\_INCORRECT\_-WARNING, 28
  - NPP\_ALIGNMENT\_ERROR, 28
  - NPP\_BAD\_ARG\_ERROR, 27
  - NPP\_BOTH\_AXIS, 26
  - NPP\_CMP\_EQ, 26

- NPP\_CMP\_GREATER, 26
- NPP\_CMP\_GREATER\_EQ, 26
- NPP\_CMP\_LESS, 26
- NPP\_CMP\_LESS\_EQ, 26
- NPP\_COEFF\_ERROR, 27
- NPP\_CUDA\_1\_0, 26
- NPP\_CUDA\_1\_1, 26
- NPP\_CUDA\_1\_2, 26
- NPP\_CUDA\_1\_3, 26
- NPP\_CUDA\_2\_0, 26
- NPP\_CUDA\_KERNEL\_EXECUTION\_-  
ERROR, 28
- NPP\_CUDA\_NOT\_CAPABLE, 26
- NPP\_CUDA\_UNKNOWN\_VERSION, 26
- NPP\_DOUBLE\_SIZE\_WARNING, 28
- NPP\_ERROR, 28
- NPP\_HAAR\_CLASSIFIER\_PIXEL\_-  
MATCH\_ERROR, 27
- NPP\_HISTO\_NUMBER\_OF\_LEVELS\_-  
ERROR, 28
- NPP\_HORIZONTAL\_AXIS, 26
- NPP\_INTERPOLATION\_ERROR, 27
- NPP\_INVALID\_INPUT, 28
- NPP\_LUT\_NUMBER\_OF\_LEVELS\_-  
ERROR, 27
- NPP\_MEM\_ALLOC\_ERR, 28
- NPP\_MEMCPY\_ERROR, 28
- NPP\_MEMFREE\_ERR, 27
- NPP\_MEMSET\_ERR, 27
- NPP\_MIRROR\_FLIP\_ERR, 28
- NPP\_MISALIGNED\_DST\_ROI\_WARNING,  
28
- NPP\_NO\_ERROR, 28
- NPP\_NOT\_EVEN\_STEP\_ERROR, 27
- NPP\_NOT\_IMPLEMENTED\_ERROR, 28
- NPP\_NOT\_SUFFICIENT\_COMPUTE\_-  
CAPABILITY, 27
- NPP\_NOT\_SUPPORTED\_MODE\_ERROR,  
27
- NPP\_NULL\_POINTER\_ERROR, 28
- NPP\_ODD\_ROI\_WARNING, 28
- NPP\_POINTER\_ERROR, 28
- NPP\_QUAD\_ERROR, 27
- NPP\_RECT\_ERROR, 27
- NPP\_RESIZE\_FACTOR\_ERROR, 27
- NPP\_RESIZE\_NO\_OPERATION\_ERROR,  
27
- NPP\_RND\_FINANCIAL, 27
- NPP\_RND\_NEAR, 27
- NPP\_RND\_ZERO, 27
- NPP\_ROUND\_MODE\_NOT\_-  
SUPPORTED\_ERROR, 27
- NPP\_SIZE\_ERROR, 28
- NPP\_STEP\_ERROR, 28
- NPP\_SUCCESS, 28
- NPP\_TEXTURE\_BIND\_ERROR, 27
- NPP\_VERTICAL\_AXIS, 26
- NPP\_WARNING, 28
- NPP\_WRONG\_INTERSECTION\_QUAD\_-  
WARNING, 28
- NPP\_WRONG\_INTERSECTION\_ROI\_-  
ERROR, 27
- nppAlgHintAccurate, 26
- nppAlgHintFast, 26
- nppAlgHintNone, 26
- NPPI\_INTER\_CUBIC, 27
- NPPI\_INTER\_LANCZOS, 27
- NPPI\_INTER\_LINEAR, 27
- NPPI\_INTER\_NN, 27
- NPPI\_INTER\_SUPER, 27
- NPPI\_SMOOTH\_EDGE, 27
- typedefs\_npp
  - NPP\_MAX\_16S, 24
  - NPP\_MAX\_16U, 24
  - NPP\_MAX\_32S, 24
  - NPP\_MAX\_32U, 24
  - NPP\_MAX\_64S, 24
  - NPP\_MAX\_8S, 24
  - NPP\_MAX\_8U, 24
  - NPP\_MAXABS\_32F, 25
  - NPP\_MAXABS\_64F, 25
  - NPP\_MIN\_16S, 25
  - NPP\_MIN\_16U, 25
  - NPP\_MIN\_32S, 25
  - NPP\_MIN\_32U, 25
  - NPP\_MIN\_64S, 25
  - NPP\_MIN\_8S, 25
  - NPP\_MIN\_8U, 25
  - NPP\_MINABS\_32F, 25
  - NPP\_MINABS\_64F, 25
  - NppCmpOp, 26
  - NppGpuComputeCapability, 26
  - NppHintAlgorithm, 26
  - NppiAxis, 26
  - NppiInterpolationMode, 26
  - NppRoundMode, 27
  - NppStatus, 27
- width
  - NppiRect, 259
  - NppiSize, 260
- x
  - NppiPoint, 258
  - NppiRect, 259
- y
  - NppiPoint, 258
  - NppiRect, 259