

NVIDIA Performance Primitives (NPP)  
Version 9.0

August 18, 2017



# Contents

<b>1</b>	<b>NVIDIA Performance Primitives</b>	<b>1</b>
1.1	What is NPP? . . . . .	2
1.2	Documentation . . . . .	2
1.3	Technical Specifications . . . . .	2
1.4	Files . . . . .	3
1.4.1	Header Files . . . . .	3
1.4.2	Library Files . . . . .	3
1.5	Supported NVIDIA Hardware . . . . .	4
<b>2</b>	<b>General API Conventions</b>	<b>5</b>
2.1	Memory Management . . . . .	6
2.1.1	Scratch Buffer and Host Pointer . . . . .	6
2.2	Function Naming . . . . .	7
2.3	Integer Result Scaling . . . . .	7
2.4	Rounding Modes . . . . .	8
2.4.1	Rounding Mode Parameter . . . . .	8
<b>3</b>	<b>Signal-Processing Specific API Conventions</b>	<b>9</b>
3.1	Signal Data . . . . .	10
3.1.1	Parameter Names for Signal Data . . . . .	10
3.1.1.1	Source Signal Pointer . . . . .	10
3.1.1.2	Destination Signal Pointer . . . . .	10
3.1.1.3	In-Place Signal Pointer . . . . .	10
3.1.2	Signal Data Alignment Requirements . . . . .	11
3.1.3	Signal Data Related Error Codes . . . . .	11
3.2	Signal Length . . . . .	11
3.2.1	Length Related Error Codes . . . . .	11
<b>4</b>	<b>Imaging-Processing Specific API Conventions</b>	<b>13</b>

4.1	Function Naming	14
4.2	Image Data	14
4.2.1	Line Step	15
4.2.2	Parameter Names for Image Data	15
4.2.2.1	Passing Source-Image Data	15
4.2.2.2	Passing Destination-Image Data	16
4.2.2.3	Passing In-Place Image Data	18
4.2.2.4	Passing Mask-Image Data	18
4.2.2.5	Passing Channel-of-Interest Data	18
4.2.3	Image Data Alignment Requirements	18
4.2.4	Image Data Related Error Codes	19
4.3	Region-of-Interest (ROI)	19
4.3.1	ROI Related Error Codes	19
4.4	Masked Operation	20
4.5	Channel-of-Interest API	20
4.5.1	Select-Channel Source-Image Pointer	20
4.5.2	Select-Channel Source-Image	20
4.5.3	Select-Channel Destination-Image Pointer	20
4.6	Source-Image Sampling	21
4.6.1	Point-Wise Operations	21
4.6.2	Neighborhood Operations	21
4.6.2.1	Mask-Size Parameter	21
4.6.2.2	Anchor-Point Parameter	22
4.6.2.3	Sampling Beyond Image Boundaries	22
<b>5</b>	<b>Module Index</b>	<b>23</b>
5.1	Modules	23
<b>6</b>	<b>Data Structure Index</b>	<b>25</b>
6.1	Data Structures	25
<b>7</b>	<b>Module Documentation</b>	<b>27</b>
7.1	NPP Core	27
7.1.1	Detailed Description	28
7.1.2	Function Documentation	28
7.1.2.1	nppGetGpuComputeCapability	28
7.1.2.2	nppGetGpuDeviceProperties	28
7.1.2.3	nppGetGpuName	28

---

7.1.2.4	nppGetGpuNumSMs	28
7.1.2.5	nppGetLibVersion	29
7.1.2.6	nppGetMaxThreadsPerBlock	29
7.1.2.7	nppGetMaxThreadsPerSM	29
7.1.2.8	nppGetStream	29
7.1.2.9	nppGetStreamMaxThreadsPerSM	29
7.1.2.10	nppGetStreamNumSMs	29
7.1.2.11	nppSetStream	30
7.2	NPP Type Definitions and Constants	31
7.2.1	Define Documentation	37
7.2.1.1	NPP_HOG_MAX_BINS_PER_CELL	37
7.2.1.2	NPP_HOG_MAX_BLOCK_SIZE	37
7.2.1.3	NPP_HOG_MAX_CELL_SIZE	37
7.2.1.4	NPP_HOG_MAX_CELLS_PER_DESCRIPTOR	38
7.2.1.5	NPP_HOG_MAX_DESCRIPTOR_LOCATIONS_PER_CALL	38
7.2.1.6	NPP_HOG_MAX_OVERLAPPING_BLOCKS_PER_DESCRIPTOR	38
7.2.1.7	NPP_MAX_16S	38
7.2.1.8	NPP_MAX_16U	38
7.2.1.9	NPP_MAX_32S	38
7.2.1.10	NPP_MAX_32U	38
7.2.1.11	NPP_MAX_64S	38
7.2.1.12	NPP_MAX_64U	38
7.2.1.13	NPP_MAX_8S	38
7.2.1.14	NPP_MAX_8U	38
7.2.1.15	NPP_MAXABS_32F	39
7.2.1.16	NPP_MAXABS_64F	39
7.2.1.17	NPP_MIN_16S	39
7.2.1.18	NPP_MIN_16U	39
7.2.1.19	NPP_MIN_32S	39
7.2.1.20	NPP_MIN_32U	39
7.2.1.21	NPP_MIN_64S	39
7.2.1.22	NPP_MIN_64U	39
7.2.1.23	NPP_MIN_8S	39
7.2.1.24	NPP_MIN_8U	39
7.2.1.25	NPP_MINABS_32F	39
7.2.1.26	NPP_MINABS_64F	40

7.2.2	Enumeration Type Documentation	40
7.2.2.1	NppCmpOp	40
7.2.2.2	NppGpuComputeCapability	40
7.2.2.3	NppHintAlgorithm	41
7.2.2.4	NppiAlphaOp	41
7.2.2.5	NppiAxis	41
7.2.2.6	NppiBayerGridPosition	41
7.2.2.7	NppiBorderType	42
7.2.2.8	NppiDifferentialKernel	42
7.2.2.9	NppiHuffmanTableType	42
7.2.2.10	NppiInterpolationMode	42
7.2.2.11	NppiMaskSize	43
7.2.2.12	NppiNorm	43
7.2.2.13	NppRoundMode	43
7.2.2.14	NppStatus	44
7.2.2.15	NppsZCType	46
7.3	Basic NPP Data Types	47
7.3.1	Typedef Documentation	48
7.3.1.1	Npp16s	48
7.3.1.2	Npp16u	48
7.3.1.3	Npp32f	48
7.3.1.4	Npp32fc	48
7.3.1.5	Npp32s	48
7.3.1.6	Npp32sc	49
7.3.1.7	Npp32u	49
7.3.1.8	Npp32uc	49
7.3.1.9	Npp64f	49
7.3.1.10	Npp64fc	49
7.3.1.11	Npp64s	49
7.3.1.12	Npp64sc	49
7.3.1.13	Npp64u	49
7.3.1.14	Npp8s	49
7.3.1.15	Npp8u	49
7.3.2	Function Documentation	49
7.3.2.1	__align__	49
7.3.2.2	__align__	50

7.3.3	Variable Documentation	50
7.3.3.1	Npp16sc	50
7.3.3.2	Npp16uc	50
7.3.3.3	Npp8uc	50
7.4	Threshold and Compare Operations	51
7.4.1	Detailed Description	51
7.5	Threshold Operations	52
7.5.1	Detailed Description	66
7.5.2	Function Documentation	66
7.5.2.1	nppiThreshold_16s_AC4IR	66
7.5.2.2	nppiThreshold_16s_AC4R	66
7.5.2.3	nppiThreshold_16s_C1IR	67
7.5.2.4	nppiThreshold_16s_C1R	67
7.5.2.5	nppiThreshold_16s_C3IR	68
7.5.2.6	nppiThreshold_16s_C3R	68
7.5.2.7	nppiThreshold_16u_AC4IR	69
7.5.2.8	nppiThreshold_16u_AC4R	69
7.5.2.9	nppiThreshold_16u_C1IR	70
7.5.2.10	nppiThreshold_16u_C1R	70
7.5.2.11	nppiThreshold_16u_C3IR	70
7.5.2.12	nppiThreshold_16u_C3R	71
7.5.2.13	nppiThreshold_32f_AC4IR	71
7.5.2.14	nppiThreshold_32f_AC4R	72
7.5.2.15	nppiThreshold_32f_C1IR	72
7.5.2.16	nppiThreshold_32f_C1R	73
7.5.2.17	nppiThreshold_32f_C3IR	73
7.5.2.18	nppiThreshold_32f_C3R	74
7.5.2.19	nppiThreshold_8u_AC4IR	74
7.5.2.20	nppiThreshold_8u_AC4R	75
7.5.2.21	nppiThreshold_8u_C1IR	75
7.5.2.22	nppiThreshold_8u_C1R	76
7.5.2.23	nppiThreshold_8u_C3IR	76
7.5.2.24	nppiThreshold_8u_C3R	77
7.5.2.25	nppiThreshold_GT_16s_AC4IR	77
7.5.2.26	nppiThreshold_GT_16s_AC4R	77
7.5.2.27	nppiThreshold_GT_16s_C1IR	78

7.5.2.28	nppiThreshold_GT_16s_C1R	78
7.5.2.29	nppiThreshold_GT_16s_C3IR	79
7.5.2.30	nppiThreshold_GT_16s_C3R	79
7.5.2.31	nppiThreshold_GT_16u_AC4IR	79
7.5.2.32	nppiThreshold_GT_16u_AC4R	80
7.5.2.33	nppiThreshold_GT_16u_C1IR	80
7.5.2.34	nppiThreshold_GT_16u_C1R	81
7.5.2.35	nppiThreshold_GT_16u_C3IR	81
7.5.2.36	nppiThreshold_GT_16u_C3R	81
7.5.2.37	nppiThreshold_GT_32f_AC4IR	82
7.5.2.38	nppiThreshold_GT_32f_AC4R	82
7.5.2.39	nppiThreshold_GT_32f_C1IR	83
7.5.2.40	nppiThreshold_GT_32f_C1R	83
7.5.2.41	nppiThreshold_GT_32f_C3IR	83
7.5.2.42	nppiThreshold_GT_32f_C3R	84
7.5.2.43	nppiThreshold_GT_8u_AC4IR	84
7.5.2.44	nppiThreshold_GT_8u_AC4R	85
7.5.2.45	nppiThreshold_GT_8u_C1IR	85
7.5.2.46	nppiThreshold_GT_8u_C1R	85
7.5.2.47	nppiThreshold_GT_8u_C3IR	86
7.5.2.48	nppiThreshold_GT_8u_C3R	86
7.5.2.49	nppiThreshold_GTVVal_16s_AC4IR	87
7.5.2.50	nppiThreshold_GTVVal_16s_AC4R	87
7.5.2.51	nppiThreshold_GTVVal_16s_C1IR	87
7.5.2.52	nppiThreshold_GTVVal_16s_C1R	88
7.5.2.53	nppiThreshold_GTVVal_16s_C3IR	88
7.5.2.54	nppiThreshold_GTVVal_16s_C3R	89
7.5.2.55	nppiThreshold_GTVVal_16u_AC4IR	89
7.5.2.56	nppiThreshold_GTVVal_16u_AC4R	89
7.5.2.57	nppiThreshold_GTVVal_16u_C1IR	90
7.5.2.58	nppiThreshold_GTVVal_16u_C1R	90
7.5.2.59	nppiThreshold_GTVVal_16u_C3IR	91
7.5.2.60	nppiThreshold_GTVVal_16u_C3R	91
7.5.2.61	nppiThreshold_GTVVal_32f_AC4IR	92
7.5.2.62	nppiThreshold_GTVVal_32f_AC4R	92
7.5.2.63	nppiThreshold_GTVVal_32f_C1IR	92



---

7.5.2.64	nppiThreshold_GTVal_32f_C1R	93
7.5.2.65	nppiThreshold_GTVal_32f_C3IR	93
7.5.2.66	nppiThreshold_GTVal_32f_C3R	94
7.5.2.67	nppiThreshold_GTVal_8u_AC4IR	94
7.5.2.68	nppiThreshold_GTVal_8u_AC4R	94
7.5.2.69	nppiThreshold_GTVal_8u_C1IR	95
7.5.2.70	nppiThreshold_GTVal_8u_C1R	95
7.5.2.71	nppiThreshold_GTVal_8u_C3IR	96
7.5.2.72	nppiThreshold_GTVal_8u_C3R	96
7.5.2.73	nppiThreshold_LT_16s_AC4IR	97
7.5.2.74	nppiThreshold_LT_16s_AC4R	97
7.5.2.75	nppiThreshold_LT_16s_C1IR	97
7.5.2.76	nppiThreshold_LT_16s_C1R	98
7.5.2.77	nppiThreshold_LT_16s_C3IR	98
7.5.2.78	nppiThreshold_LT_16s_C3R	99
7.5.2.79	nppiThreshold_LT_16u_AC4IR	99
7.5.2.80	nppiThreshold_LT_16u_AC4R	99
7.5.2.81	nppiThreshold_LT_16u_C1IR	100
7.5.2.82	nppiThreshold_LT_16u_C1R	100
7.5.2.83	nppiThreshold_LT_16u_C3IR	101
7.5.2.84	nppiThreshold_LT_16u_C3R	101
7.5.2.85	nppiThreshold_LT_32f_AC4IR	101
7.5.2.86	nppiThreshold_LT_32f_AC4R	102
7.5.2.87	nppiThreshold_LT_32f_C1IR	102
7.5.2.88	nppiThreshold_LT_32f_C1R	103
7.5.2.89	nppiThreshold_LT_32f_C3IR	103
7.5.2.90	nppiThreshold_LT_32f_C3R	103
7.5.2.91	nppiThreshold_LT_8u_AC4IR	104
7.5.2.92	nppiThreshold_LT_8u_AC4R	104
7.5.2.93	nppiThreshold_LT_8u_C1IR	105
7.5.2.94	nppiThreshold_LT_8u_C1R	105
7.5.2.95	nppiThreshold_LT_8u_C3IR	105
7.5.2.96	nppiThreshold_LT_8u_C3R	106
7.5.2.97	nppiThreshold_LTVAl_16s_AC4IR	106
7.5.2.98	nppiThreshold_LTVAl_16s_AC4R	107
7.5.2.99	nppiThreshold_LTVAl_16s_C1IR	107

7.5.2.100 nppiThreshold_LTVVal_16s_C1R	107
7.5.2.101 nppiThreshold_LTVVal_16s_C3IR	108
7.5.2.102 nppiThreshold_LTVVal_16s_C3R	108
7.5.2.103 nppiThreshold_LTVVal_16u_AC4IR	109
7.5.2.104 nppiThreshold_LTVVal_16u_AC4R	109
7.5.2.105 nppiThreshold_LTVVal_16u_C1IR	110
7.5.2.106 nppiThreshold_LTVVal_16u_C1R	110
7.5.2.107 nppiThreshold_LTVVal_16u_C3IR	110
7.5.2.108 nppiThreshold_LTVVal_16u_C3R	111
7.5.2.109 nppiThreshold_LTVVal_32f_AC4IR	111
7.5.2.110 nppiThreshold_LTVVal_32f_AC4R	112
7.5.2.111 nppiThreshold_LTVVal_32f_C1IR	112
7.5.2.112 nppiThreshold_LTVVal_32f_C1R	112
7.5.2.113 nppiThreshold_LTVVal_32f_C3IR	113
7.5.2.114 nppiThreshold_LTVVal_32f_C3R	113
7.5.2.115 nppiThreshold_LTVVal_8u_AC4IR	114
7.5.2.116 nppiThreshold_LTVVal_8u_AC4R	114
7.5.2.117 nppiThreshold_LTVVal_8u_C1IR	115
7.5.2.118 nppiThreshold_LTVVal_8u_C1R	115
7.5.2.119 nppiThreshold_LTVVal_8u_C3IR	115
7.5.2.120 nppiThreshold_LTVVal_8u_C3R	116
7.5.2.121 nppiThreshold_LTVValGTVal_16s_AC4IR	116
7.5.2.122 nppiThreshold_LTVValGTVal_16s_AC4R	117
7.5.2.123 nppiThreshold_LTVValGTVal_16s_C1IR	117
7.5.2.124 nppiThreshold_LTVValGTVal_16s_C1R	118
7.5.2.125 nppiThreshold_LTVValGTVal_16s_C3IR	118
7.5.2.126 nppiThreshold_LTVValGTVal_16s_C3R	119
7.5.2.127 nppiThreshold_LTVValGTVal_16u_AC4IR	119
7.5.2.128 nppiThreshold_LTVValGTVal_16u_AC4R	120
7.5.2.129 nppiThreshold_LTVValGTVal_16u_C1IR	120
7.5.2.130 nppiThreshold_LTVValGTVal_16u_C1R	121
7.5.2.131 nppiThreshold_LTVValGTVal_16u_C3IR	121
7.5.2.132 nppiThreshold_LTVValGTVal_16u_C3R	122
7.5.2.133 nppiThreshold_LTVValGTVal_32f_AC4IR	122
7.5.2.134 nppiThreshold_LTVValGTVal_32f_AC4R	123
7.5.2.135 nppiThreshold_LTVValGTVal_32f_C1IR	123

7.5.2.136	<code>nppiThreshold_LTValGTVal_32f_C1R</code>	124
7.5.2.137	<code>nppiThreshold_LTValGTVal_32f_C3IR</code>	124
7.5.2.138	<code>nppiThreshold_LTValGTVal_32f_C3R</code>	125
7.5.2.139	<code>nppiThreshold_LTValGTVal_8u_AC4IR</code>	125
7.5.2.140	<code>nppiThreshold_LTValGTVal_8u_AC4R</code>	126
7.5.2.141	<code>nppiThreshold_LTValGTVal_8u_C1IR</code>	126
7.5.2.142	<code>nppiThreshold_LTValGTVal_8u_C1R</code>	127
7.5.2.143	<code>nppiThreshold_LTValGTVal_8u_C3IR</code>	127
7.5.2.144	<code>nppiThreshold_LTValGTVal_8u_C3R</code>	128
7.5.2.145	<code>nppiThreshold_Val_16s_AC4IR</code>	128
7.5.2.146	<code>nppiThreshold_Val_16s_AC4R</code>	129
7.5.2.147	<code>nppiThreshold_Val_16s_C1IR</code>	129
7.5.2.148	<code>nppiThreshold_Val_16s_C1R</code>	130
7.5.2.149	<code>nppiThreshold_Val_16s_C3IR</code>	130
7.5.2.150	<code>nppiThreshold_Val_16s_C3R</code>	131
7.5.2.151	<code>nppiThreshold_Val_16u_AC4IR</code>	131
7.5.2.152	<code>nppiThreshold_Val_16u_AC4R</code>	132
7.5.2.153	<code>nppiThreshold_Val_16u_C1IR</code>	132
7.5.2.154	<code>nppiThreshold_Val_16u_C1R</code>	133
7.5.2.155	<code>nppiThreshold_Val_16u_C3IR</code>	133
7.5.2.156	<code>nppiThreshold_Val_16u_C3R</code>	134
7.5.2.157	<code>nppiThreshold_Val_32f_AC4IR</code>	134
7.5.2.158	<code>nppiThreshold_Val_32f_AC4R</code>	135
7.5.2.159	<code>nppiThreshold_Val_32f_C1IR</code>	135
7.5.2.160	<code>nppiThreshold_Val_32f_C1R</code>	136
7.5.2.161	<code>nppiThreshold_Val_32f_C3IR</code>	136
7.5.2.162	<code>nppiThreshold_Val_32f_C3R</code>	137
7.5.2.163	<code>nppiThreshold_Val_8u_AC4IR</code>	137
7.5.2.164	<code>nppiThreshold_Val_8u_AC4R</code>	138
7.5.2.165	<code>nppiThreshold_Val_8u_C1IR</code>	138
7.5.2.166	<code>nppiThreshold_Val_8u_C1R</code>	139
7.5.2.167	<code>nppiThreshold_Val_8u_C3IR</code>	139
7.5.2.168	<code>nppiThreshold_Val_8u_C3R</code>	140
7.6	Compare Operations	141
7.6.1	Detailed Description	144
7.6.2	Function Documentation	144

7.6.2.1	nppiCompare_16s_AC4R	144
7.6.2.2	nppiCompare_16s_C1R	145
7.6.2.3	nppiCompare_16s_C3R	145
7.6.2.4	nppiCompare_16s_C4R	146
7.6.2.5	nppiCompare_16u_AC4R	146
7.6.2.6	nppiCompare_16u_C1R	147
7.6.2.7	nppiCompare_16u_C3R	147
7.6.2.8	nppiCompare_16u_C4R	148
7.6.2.9	nppiCompare_32f_AC4R	148
7.6.2.10	nppiCompare_32f_C1R	149
7.6.2.11	nppiCompare_32f_C3R	149
7.6.2.12	nppiCompare_32f_C4R	150
7.6.2.13	nppiCompare_8u_AC4R	150
7.6.2.14	nppiCompare_8u_C1R	151
7.6.2.15	nppiCompare_8u_C3R	151
7.6.2.16	nppiCompare_8u_C4R	152
7.6.2.17	nppiCompareC_16s_AC4R	152
7.6.2.18	nppiCompareC_16s_C1R	152
7.6.2.19	nppiCompareC_16s_C3R	153
7.6.2.20	nppiCompareC_16s_C4R	153
7.6.2.21	nppiCompareC_16u_AC4R	154
7.6.2.22	nppiCompareC_16u_C1R	154
7.6.2.23	nppiCompareC_16u_C3R	155
7.6.2.24	nppiCompareC_16u_C4R	155
7.6.2.25	nppiCompareC_32f_AC4R	155
7.6.2.26	nppiCompareC_32f_C1R	156
7.6.2.27	nppiCompareC_32f_C3R	156
7.6.2.28	nppiCompareC_32f_C4R	157
7.6.2.29	nppiCompareC_8u_AC4R	157
7.6.2.30	nppiCompareC_8u_C1R	158
7.6.2.31	nppiCompareC_8u_C3R	158
7.6.2.32	nppiCompareC_8u_C4R	158
7.6.2.33	nppiCompareEqualEps_32f_AC4R	159
7.6.2.34	nppiCompareEqualEps_32f_C1R	159
7.6.2.35	nppiCompareEqualEps_32f_C3R	160
7.6.2.36	nppiCompareEqualEps_32f_C4R	160

7.6.2.37	nppiCompareEqualEpsC_32f_AC4R	161
7.6.2.38	nppiCompareEqualEpsC_32f_C1R	161
7.6.2.39	nppiCompareEqualEpsC_32f_C3R	162
7.6.2.40	nppiCompareEqualEpsC_32f_C4R	162
<b>8</b>	<b>Data Structure Documentation</b>	<b>163</b>
8.1	NPP_ALIGN_16 Struct Reference	163
8.1.1	Detailed Description	163
8.1.2	Field Documentation	163
8.1.2.1	im	163
8.1.2.2	im	164
8.1.2.3	re	164
8.1.2.4	re	164
8.2	NPP_ALIGN_8 Struct Reference	165
8.2.1	Detailed Description	165
8.2.2	Field Documentation	165
8.2.2.1	im	165
8.2.2.2	im	165
8.2.2.3	im	165
8.2.2.4	re	166
8.2.2.5	re	166
8.2.2.6	re	166
8.3	NppiHaarBuffer Struct Reference	167
8.3.1	Field Documentation	167
8.3.1.1	haarBuffer	167
8.3.1.2	haarBufferSize	167
8.4	NppiHaarClassifier_32f Struct Reference	168
8.4.1	Field Documentation	168
8.4.1.1	classifiers	168
8.4.1.2	classifierSize	168
8.4.1.3	classifierStep	168
8.4.1.4	counterDevice	168
8.4.1.5	numClassifiers	168
8.5	NppiHOGConfig Struct Reference	169
8.5.1	Detailed Description	169
8.5.2	Field Documentation	169
8.5.2.1	cellSize	169

---

8.5.2.2	detectionWindowSize	169
8.5.2.3	histogramBlockSize	169
8.5.2.4	nHistogramBins	169
8.6	NppiPoint Struct Reference	170
8.6.1	Detailed Description	170
8.6.2	Field Documentation	170
8.6.2.1	x	170
8.6.2.2	y	170
8.7	NppiRect Struct Reference	171
8.7.1	Detailed Description	171
8.7.2	Field Documentation	171
8.7.2.1	height	171
8.7.2.2	width	171
8.7.2.3	x	171
8.7.2.4	y	171
8.8	NppiSize Struct Reference	172
8.8.1	Detailed Description	172
8.8.2	Field Documentation	172
8.8.2.1	height	172
8.8.2.2	width	172
8.9	NppLibraryVersion Struct Reference	173
8.9.1	Field Documentation	173
8.9.1.1	build	173
8.9.1.2	major	173
8.9.1.3	minor	173
8.10	NppPointPolar Struct Reference	174
8.10.1	Detailed Description	174
8.10.2	Field Documentation	174
8.10.2.1	rho	174
8.10.2.2	theta	174

# Chapter 1

## NVIDIA Performance Primitives

Note: The static NPP libraries depend on a common thread abstraction layer library called cuLIBOS (lib-culibos.a) that is now distributed as part of the toolkit. Consequently, cuLIBOS must be provided to the linker when the static library is being linked against. To minimize library loading and CUDA runtime startup times it is recommended to use the static library(s) whenever possible. To improve loading and runtime performance when using dynamic libraries, NPP 9.0 has deprecated the full sized nppi library and replaced it with a full set of nppi sub-libraries. Linking to only the sub-libraries that contain functions that your application uses can significantly improve load time and runtime startup performance. Some nppi functions make calls to other nppi and/or npps functions internally so you may need to link to a few extra libraries depending on what function calls your application makes. The nppi sub-libraries are split into sections corresponding to the way that nppi header files are split. This list of sub-libraries is as follows:

```
nppial arithmetic and logical operation functions in nppi_arithmetic_and_logical_operations.h
nppicc color conversion and sampling functions in nppi_color_conversion.h
nppicom JPEG compression and decompression functions in nppi_compression_functions.h
nppidei data exchange and initialization functions in nppi_data_exchange_and_initialization.h
nppif filtering and computer vision functions in nppi_filter_functions.h
nppig geometry transformation functions found in nppi_geometry_transforms.h
nppim morphological operation functions found in nppi_morphological_operations.h
nppist statistics and linear transform in nppi_statistics_functions.h and nppi_linear_transforms.h
nppisu memory support functions in nppi_support_functions.h
nppitc threshold and compare operation functions in nppi_threshold_and_compare_operations.h
```

For example, on Linux, to compile a small application foo using NPP against the dynamic library, the following command can be used:

```
nvcc foo.c -lnppi -o foo
```

Whereas to compile against the static NPP library, the following command has to be used:

```
nvcc foo.c -lnppi_static -lculibos -o foo
```

It is also possible to use the native host C++ compiler. Depending on the host operating system, some additional libraries like pthread or dl might be needed on the linking line. The following command on Linux is suggested:

```
g++ foo.c -lnppi_static -lculibos -lcudart_static -lpthread -ldl
-I <cuda-toolkit-path>/include -L <cuda-toolkit-path>/lib64 -o foo
```

NPP is a stateless API, as of NPP 6.5 the ONLY state that NPP remembers between function calls is the current stream ID, i.e. the stream ID that was set in the most recent nppSetStream call and a few bits

of device specific information about that stream. The default stream ID is 0. If an application intends to use NPP with multiple streams then it is the responsibility of the application to call `nppSetStream` whenever it wishes to change stream IDs. Several NPP functions may call other NPP functions internally to complete their functionality. For this reason it is recommended that `cudaDeviceSynchronize` (or at least `cudaStreamSynchronize`) be called before making an `nppSetStream` call to change to a new stream ID. This will insure that any internal function calls that have not yet occurred will be completed using the current stream ID before it changes to a new ID. Calling `cudaDeviceSynchronize` frequently call kill performance so minimizing the frequency of these calls is critical for good performance. It is not necessary to call `cudaDeviceSynchronize` for stream management while the same stream ID is used for multiple NPP calls. All NPP functions should be thread safe except for the following functions:

```
nppiDCTQuantFwd8x8LS_JPEG_8u16s_C1R  
nppiDCTQuantInv8x8LS_JPEG_16s8u_C1R
```

## 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, 8, and 10 (64-bit and 32-bit)
- Microsoft Windows Vista (64-bit and 32-bit)
- Linux (Centos, Ubuntu, and several others) (64-bit and 32-bit)
- Mac OS X (64-bit)
- Android on Arm (32-bit and 64-bit)



## 1.4 Files

NPP is comprised of the following files:

### 1.4.1 Header Files

- [nppdefs.h](#)
- [nppcore.h](#)
- [nppi.h](#)
- [npps.h](#)
- [nppversion.h](#)
- [npp.h](#)

All those header files are located in the CUDA Toolkit's

```
/include/
```

directory.

### 1.4.2 Library Files

Starting with Version 5.5 NPP's functionality is now split up into 3 distinct library groups:

- A core library (NPPC) containing basic functionality from the `npp.h` header files as well as functionality shared by the other two libraries.
- The image processing library NPPI. Any functions from the `nppi.h` header file (or the various header files named "`nppi_XXX.h`") are bundled into the NPPI library.
- The signal processing library NPPS. Any function from the `npps.h` header file (or the various header files named "`npps_XXX.h`") are bundled into the NPPS library.

On the Windows platform the NPP stub libraries are found in the CUDA Toolkit's library directory:

```
/lib/nppc.lib
```

```
/lib/nppial.lib
```

```
/lib/nppicc.lib
```

```
/lib/nppicom.lib
```

```
/lib/nppidei.lib
```

```
/lib/nppif.lib
```

```
/lib/nppig.lib
```

```
/lib/nppim.lib
```

```
/lib/nppist.lib
```

```
/lib/nppisu.lib
```

```
/lib/nppitc.lib
```

```
/lib/npps.lib
```

The matching DLLs are located in the CUDA Toolkit's binary directory. Example

```
/bin/nppial64_90_<build_no>.dll // Dynamic image-processing library for 64-bit Windows.
```

On Linux and Mac platforms the dynamic libraries are located in the lib directory

```
/lib/libnppc.so.9.0.<build_no> // NPP dynamic core library for Linux
```

```
/lib/libnpps.9.0.dylib // NPP dynamic signal processing library for Mac
```

## 1.5 Supported NVIDIA Hardware

NPP runs on all CUDA capable NVIDIA hardware. For details please see [http://www.nvidia.com/object/cuda\\_learn\\_products.html](http://www.nvidia.com/object/cuda_learn_products.html)

## **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(...)
```

### 2.1.1 Scratch Buffer and Host Pointer

Some primitives of NPP require additional device memory buffers (scratch buffers) for calculations, e.g. signal and image reductions (Sum, Max, Min, MinMax, etc.). In order to give the NPP user maximum control regarding memory allocations and performance, it is the user's responsibility to allocate and delete those temporary buffers. For one this has the benefit that the library will not allocate memory unbeknownst to the user. It also allows developers who invoke the same primitive repeatedly to allocate the scratch only once, improving performance and potential device-memory fragmentation.

Scratch-buffer memory is unstructured and may be passed to the primitive in uninitialized form. This allows for reuse of the same scratch buffers with any primitive require scratch memory, as long as it is sufficiently sized.

The minimum scratch-buffer size for a given primitive (e.g. `nppsSum_32f()`) can be obtained by a companion function (e.g. `nppsSumGetBufferSize_32f()`). The buffer size is returned via a host pointer as allocation of the scratch-buffer is performed via CUDA runtime host code.

An example to invoke signal sum primitive and allocate and free the necessary scratch memory:

```
// pSrc, pSum, pDeviceBuffer are all device pointers.
Npp32f * pSrc;
Npp32f * pSum;
Npp8u * pDeviceBuffer;
int nLength = 1024;

// Allocate the device memroy.
cudaMalloc((void **)(&pSrc), sizeof(Npp32f) * nLength);
nppsSet_32f(1.0f, pSrc, nLength);
cudaMalloc((void **)(&pSum), sizeof(Npp32f) * 1);

// Compute the appropriate size of the scratch-memory buffer
int nBufferSize;
nppsSumGetBufferSize_32f(nLength, &nBufferSize);
// Allocate the scratch buffer
cudaMalloc((void **)(&pDeviceBuffer), nBufferSize);

// Call the primitive with the scratch buffer
```

```

nppsSum_32f(pSrc, nLength, pSum, pDeviceBuffer);
Npp32f nSumHost;
cudaMemcpy(&nSumHost, pSum, sizeof(Npp32f) * 1, cudaMemcpyDeviceToHost);
printf("sum = %f\n", nSumHost); // nSumHost = 1024.0f;

// Free the device memory
cudaFree(pSrc);
cudaFree(pDeviceBuffer);
cudaFree(pSum);

```

## 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.

## 2.3 Integer Result Scaling

NPP signal processing and imaging primitives often operate on integer data. This integer data is usually a fixed point fractional representation of some physical magnitue (e.g. luminance). Because of this fixed-point nature of the representation many numerical operations (e.g. addition or multiplication) tend to produce results exceeding the original fixed-point range if treated as regular integers.

In cases where the results exceed the original range, these functions clamp the result values back to the valid range. E.g. the maximum positive value for a 16-bit unsigned integer is 32767. A multiplication operation of  $4 * 10000 = 40000$  would exceed this range. The result would be clamped to be 32767.

To avoid the level of lost information due to clamping most integer primitives allow for result scaling. Primitives with result scaling have the "Sfs" suffix in their name and provide a parameter "nScaleFactor" that controls the amount of scaling. Before the results of an operation are clamped to the valid output-data range by multiplying them with  $2^{-nScaleFactor}$ .

Example: The primitive `nppsSqr_8u_Sfs()` computes the square of 8-bit unsigned sample values in a signal (1D array of values). The maximum value of a 8-bit value is 255. The square of  $255^2 = 65025$  which would be clamped to 255 if no result scaling is performed. In order to map the maximum value of 255 to 255 in the result, one would specify an integer result scaling factor of 8, i.e. multiply each result with  $2^{-8} = \frac{1}{2^8} = \frac{1}{256}$ . The final result for a signal value of 255 being squared and scaled would be:

$$255^2 \cdot 2^{-8} = 254.00390625$$

which would be rounded to a final result of 254.

A medium gray value of 128 would result in

$$128^2 * 2^{-8} = 64$$

## 2.4 Rounding Modes

Many NPP functions require converting floating-point values to integers. The [NppRoundMode](#) enum lists NPP's supported rounding modes. Not all primitives in NPP that perform rounding as part of their functionality allow the user to specify the round-mode used. Instead they use NPP's default rounding mode, which is [NPP\\_RND\\_FINANCIAL](#).

### 2.4.1 Rounding Mode Parameter

A subset of NPP functions performing rounding as part of their functionality do allow the user to specify which rounding mode is used through a parameter of the [NppRoundMode](#) type.

## **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-Planar-Image Pointer Array

The planar source image data is generally passed via an array of pointers named

```
pSrc[]
```

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

```
nppiPrimitive_8u_P3R(const Npp8u * const pSrc[3], ...)
```

Each pointer in the array points to a different image plane.

#### 4.2.2.1.3 Source-Planar-Image Pointer

The multiple plane source image data is passed via a set of pointers named

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

The planar source image pointer is generally defined as one of a set of constant pointers with each pointer pointing to a different input image plane.

#### 4.2.2.1.4 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, ...
```

#### 4.2.2.1.5 Source-Planar-Image Line Step Array

The source planar image line step array is an array where each element of the array contains the number of bytes between successive rows for a particular plane in the input image. The source planar image line step array parameter is

```
rSrcStep[]
```

#### 4.2.2.1.6 Source-Planar-Image Line Step

The source planar image line step is the number of bytes between successive rows in a particular plane of the multiplane input image. The source planar image line step parameter is

```
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 generates multiple images as outputs the destination pointers are numbered like this:

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

#### 4.2.2.2.2 Destination-Planar-Image Pointer Array

The planar destination image data pointers are generally passed via an array of pointers named

```
pDst[]
```

Each pointer in the array points to a different image plane.

#### 4.2.2.2.3 Destination-Planar-Image Pointer

The destination planar image data is generally passed via a pointer to each plane of a multiplane output image named

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

#### 4.2.2.2.4 Destination-Image Line Step

The destination image line step parameter is

```
nDstStep
```

or in the case of multiple destination images

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

#### 4.2.2.2.5 Destination-Planar-Image Line Step Array

The destination planar image line step array is an array where each element of the array contains the number of bytes between successive rows for a particular plane in the output image. The destination planar image line step array parameter is

```
rDstStep[]
```

#### 4.2.2.2.6 Destination-Planar-Image Line Step

The destination planar image line step is the number of bytes between successive rows for a particular plane in a multiplane output image. The destination planar image line step parameter is

```
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-Image Line Step

The in-place line step parameter is

```
nSrcDstStep
```

### 4.2.2.4 Passing Mask-Image Data

Some image processing primitives have variants supporting [Masked Operation](#).

#### 4.2.2.4.1 Mask-Image Pointer

The mask-image data is generally passed via a pointer named

```
pMask
```

#### 4.2.2.4.2 Mask-Image Line Step

The mask-image line step parameter is

```
nMaskStep
```

### 4.2.2.5 Passing Channel-of-Interest Data

Some image processing primitives support [Channel-of-Interest API](#).

#### 4.2.2.5.1 Channel\_of\_Interest Number

The channel-of-interest data is generally an integer (either 1, 2, or 3):

```
nCOI
```

## 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. In most cases the ROI is passed as a single `NppiSize` struct, which provides the width and height of the ROI. This raises the 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-left corner (lowest memory address), the user simply offsets the image-data pointers to point to the first pixel of the ROI.

In practice this means that for an image (`pSrc`, `nSrcStep`) and the start-pixel of the ROI being at location (`x`, `y`), one would pass

```
pSrcOffset = pSrc + y * nSrcStep + x * 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  $(\text{widthROI} * \text{PixelSize}) > \text{nLinStep}$  indicates an error.

## 4.4 Masked Operation

Some primitive support masked operation. An "M" in the suffix of those variants indicates masked operation. Primitives supporting masked operation consume an additional input image provided via a [Mask-Image Pointer](#) and [Mask-Image Line Step](#). The mask image is interpreted by these primitives as a boolean image. The values of type `Npp8u` are interpreted as boolean values where a values of 0 indicates false, any non-zero values true.

Unless otherwise indicated the operation is only performed on pixels where its spatially corresponding mask pixel is true (non-zero). E.g. a masked copy operation would only copy those pixels in the ROI that have corresponding non-zero mask pixels.

## 4.5 Channel-of-Interest API

Some primitives allow restricting operations to a single channel of interest within a multi-channel image. These primitives are suffixed with the letter "C" (after the channel information, e.g. `nppiCopy_8u_C3CR(...)`). The channel-of-interest is generally selected by offsetting the image-data pointer to point directly to the channel- of-interest rather than the base of the first pixel in the ROI. Some primitives also explicitly specify the selected channel number and pass it via an integer, e.g. `nppiMean_StdDev_8u_C3CR(...)`.

### 4.5.1 Select-Channel Source-Image Pointer

This is a pointer to the channel-of-interest within the first pixel of the source image. E.g. if `pSrc` is the pointer to the first pixel inside the ROI of a three channel image. Using the appropriate select-channel copy primitive one could copy the second channel of this source image into the first channel of a destination image given by `pDst` by offsetting the pointer by one:

```
nppiCopy_8u_C3CR(pSrc + 1, nSrcStep, pDst, nDstStep, oSizeROI);
```

### 4.5.2 Select-Channel Source-Image

Some primitives allow the user to select the channel-of-interest by specifying the channel number (`nCOI`). This approach is typically used in the image statistical functions. For example,

```
nppiMean_StdDev_8u_C3CR(pSrc, nSrcStep, oSizeROI, nCOI, pDeviceBuffer, pMean, pStdDev );
```

The channel-of-interest number can be either 1, 2, or 3.

### 4.5.3 Select-Channel Destination-Image Pointer

This is a pointer to the channel-of-interest within the first pixel of the destination image. E.g. if `pDst` is the pointer to the first pixel inside the ROI of a three channel image. Using the appropriate select-channel

copy primitive one could copy data into the second channel of this destination image from the first channel of a source image given by pSrc by offsetting the destination pointer by one:

```
nppiCopy_8u_C3CR(pSrc, nSrcStep, pDst + 1, nDstStep, oSizeROI);
```

## 4.6 Source-Image Sampling

A large number of NPP image-processing functions consume at least one source image and produce an output image (e.g. `nppiAddC_8u_C1RSfs()` or `nppiFilterBox_8u_C1R()`). All NPP functions falling into this category also operate on ROIs (see [Region-of-Interest \(ROI\)](#)) which for these functions should be considered to describe the destination ROI. In other words the ROI describes a rectangular region in the destination image and all pixels inside of this region are being written by the function in question.

In order to use such functions successfully it is important to understand how the user defined destination ROI affects which pixels in the input image(s) are being read by the algorithms. To simplify the discussion of ROI propagation (i.e. given a destination ROI, what are the ROIs in the source(s)), it makes sense to distinguish two major cases:

1. Point-Wise Operations: These are primitives like `nppiAddC_8u_C1RSfs()`. Each output pixel requires exactly one input pixel to be read.
2. Neighborhood Operations: These are primitives like `nppiFilterBox_8u_C1R()`, which require a group of pixels from the source image(s) to be read in order to produce a single output.

### 4.6.1 Point-Wise Operations

As mentioned above, point-wise operations consume a single pixel from the input image (or a single pixel from each input image, if the operation in question has more than one input image) in order to produce a single output pixel.

### 4.6.2 Neighborhood Operations

In the case of neighborhood operations a number of input pixels (a "neighborhood" of pixels) is read in the input image (or images) in order to compute a single output pixel. All of the functions for `image_filtering_functions` and `image_morphological_operations` are neighborhood operations.

Most of these functions have parameters that affect the size and relative location of the neighborhood: a mask-size structure and an anchor-point structure. Both parameters are described in more detail in the next subsections.

#### 4.6.2.1 Mask-Size Parameter

Many NPP neighborhood operations allow the user to specify the size of the neighborhood via a parameter usually named `oMaskSize` of type `NppiSize`. In those cases the neighborhood of pixels read from the source(s) is exactly the size of the mask. Assuming the mask is anchored at location (0, 0) (see [Anchor-Point Parameter](#) below) and has a size of (w, h), i.e.

```
assert(oMaskSize.w == w);
assert(oMaskSize.h == h);
assert(oAnchor.x == 0);
assert(oAnchor.y == 0);
```

a neighborhood operation would read the following source pixels in order to compute destination pixel  $D_{i,j}$ :

$$\begin{array}{cccc} S_{i,j} & S_{i,j+1} & \cdots & S_{i,j+w-1} \\ S_{i+1,j} & S_{i+1,j+1} & \cdots & S_{i+1,j+w-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{i+h-1,j} & S_{i+h-1,j+1} & \cdots & S_{i+h-1,j+w-1} \end{array}$$

#### 4.6.2.2 Anchor-Point Parameter

Many NPP primitives performing neighborhood operations allow the user to specify the relative location of the neighborhood via a parameter usually named `oAnchor` of type [NppiPoint](#). Using the anchor a developer can choose the position of the mask (see [Mask-Size Parameter](#)) relative to current pixel index.

Using the same example as in [Mask-Size Parameter](#), but this time with an anchor position of (a, b):

```
assert(oMaskSize.w == w);
assert(oMaskSize.h == h);
assert(oAnchor.x == a);
assert(oAnchor.y == b);
```

the following pixels from the source image would be read:

$$\begin{array}{cccc} S_{i-a,j-b} & S_{i-a,j-b+1} & \cdots & S_{i-a,j-b+w-1} \\ S_{i-a+1,j-b} & S_{i-a+1,j-b+1} & \cdots & S_{i-a+1,j-b+w-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{i-a+h-1,j-b} & S_{i-a+h-1,j-b+1} & \cdots & S_{i-a+h-1,j-b+w-1} \end{array}$$

#### 4.6.2.3 Sampling Beyond Image Boundaries

NPP primitives in general and NPP neighborhood operations in particular require that all pixel locations read and written are valid and within the boundaries of the respective images. Sampling outside of the defined image data regions results in undefined behavior and may lead to system instability.

This poses a problem in practice: when processing full-size images one cannot choose the destination ROI to be the same size as the source image. Because neighborhood operations read pixels from an enlarged source ROI, the destination ROI must be shrunk so that the expanded source ROI does not exceed the source image's size.

For cases where this "shrinking" of the destination image size is unacceptable, NPP provides a set of border-expanding Copy primitives. E.g. `nppiCopyConstBorder_8u_C1R()`, `nppiCopyReplicateBorder_8u_C1R()` and `nppiCopyWrapBorder_8u_C1R()`. The user can use these primitives to "expand" the source image's size using one of the three expansion modes. The expanded image can then be safely passed to a neighborhood operation producing a full-size result.

# Chapter 5

## Module Index

### 5.1 Modules

Here is a list of all modules:

NPP Core . . . . .	27
NPP Type Definitions and Constants . . . . .	31
Basic NPP Data Types . . . . .	47
Threshold and Compare Operations . . . . .	51
Threshold Operations . . . . .	52
Compare Operations . . . . .	141



# Chapter 6

## Data Structure Index

### 6.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">NPP_ALIGN_16</a> (Complex Number This struct represents a long long complex number ) . . . .	163
<a href="#">NPP_ALIGN_8</a> (Complex Number This struct represents an unsigned int complex number ) . .	165
<a href="#">NppiHaarBuffer</a> . . . . .	167
<a href="#">NppiHaarClassifier_32f</a> . . . . .	168
<a href="#">NppiHOGConfig</a> (The <a href="#">NppiHOGConfig</a> structure defines the configuration parameters for the HOG descriptor: ) . . . . .	169
<a href="#">NppiPoint</a> (2D Point ) . . . . .	170
<a href="#">NppiRect</a> (2D Rectangle This struct contains position and size information of a rectangle in two space ) . . . . .	171
<a href="#">NppiSize</a> (2D Size This struct typically represents the size of a a rectangular region in two space )	172
<a href="#">NppLibraryVersion</a> . . . . .	173
<a href="#">NppPointPolar</a> (2D Polar Point ) . . . . .	174





# 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](#) (void)  
*Get the NPP library version.*
- [NppGpuComputeCapability](#) [nppGetGpuComputeCapability](#) (void)  
*What CUDA compute model is supported by the active CUDA device?*
- int [nppGetGpuNumSMs](#) (void)  
*Get the number of Streaming Multiprocessors (SM) on the active CUDA device.*
- int [nppGetMaxThreadsPerBlock](#) (void)  
*Get the maximum number of threads per block on the active CUDA device.*
- int [nppGetMaxThreadsPerSM](#) (void)  
*Get the maximum number of threads per SM for the active GPU.*
- int [nppGetGpuDeviceProperties](#) (int \*pMaxThreadsPerSM, int \*pMaxThreadsPerBlock, int \*pNumberOfSMs)  
*Get the maximum number of threads per SM, maximum threads per block, and number of SMs for the active GPU.*
- const char \* [nppGetGpuName](#) (void)  
*Get the name of the active CUDA device.*
- cudaStream\_t [nppGetStream](#) (void)  
*Get the NPP CUDA stream.*
- unsigned int [nppGetStreamNumSMs](#) (void)  
*Get the number of SMs on the device associated with the current NPP CUDA stream.*

- unsigned int `nppGetStreamMaxThreadsPerSM` (void)  
*Get the maximum number of threads per SM on the device associated with the current NPP CUDA stream.*
- void `nppSetStream` (cudaStream\_t hStream)  
*Set the NPP CUDA stream.*

### 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 (void)

What CUDA compute model is supported by the active 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 int nppGetGpuDeviceProperties (int \* pMaxThreadsPerSM, int \* pMaxThreadsPerBlock, int \* pNumberOfSMs)

Get the maximum number of threads per SM, maximum threads per block, and number of SMs for the active GPU.

**Returns:**

cudaSuccess for success, -1 for failure

#### 7.1.2.3 const char\* nppGetGpuName (void)

Get the name of the active CUDA device.

**Returns:**

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

#### 7.1.2.4 int nppGetGpuNumSMs (void)

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

**Returns:**

Number of SMs of the default CUDA device.

**7.1.2.5 const NppLibraryVersion\* nppGetLibVersion (void)**

Get the NPP library version.

**Returns:**

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

**7.1.2.6 int nppGetMaxThreadsPerBlock (void)**

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

**Returns:**

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

**7.1.2.7 int nppGetMaxThreadsPerSM (void)**

Get the maximum number of threads per SM for the active GPU.

**Returns:**

Maximum number of threads per SM for the active GPU

**7.1.2.8 cudaStream\_t nppGetStream (void)**

Get the NPP CUDA stream.

NPP enables concurrent device tasks via a global stream state variable. The NPP stream by default is set to stream 0, i.e. non-concurrent mode. A user can set the NPP stream to any valid CUDA stream. All CUDA commands issued by NPP (e.g. kernels launched by the NPP library) are then issued to that NPP stream.

**7.1.2.9 unsigned int nppGetStreamMaxThreadsPerSM (void)**

Get the maximum number of threads per SM on the device associated with the current NPP CUDA stream.

NPP enables concurrent device tasks via a global stream state variable. The NPP stream by default is set to stream 0, i.e. non-concurrent mode. A user can set the NPP stream to any valid CUDA stream. All CUDA commands issued by NPP (e.g. kernels launched by the NPP library) are then issued to that NPP stream. This call avoids a cudaGetDeviceProperties() call.

**7.1.2.10 unsigned int nppGetStreamNumSMs (void)**

Get the number of SMs on the device associated with the current NPP CUDA stream.

NPP enables concurrent device tasks via a global stream state variable. The NPP stream by default is set to stream 0, i.e. non-concurrent mode. A user can set the NPP stream to any valid CUDA stream. All CUDA commands issued by NPP (e.g. kernels launched by the NPP library) are then issued to that NPP stream. This call avoids a cudaGetDeviceProperties() call.

**7.1.2.11 void nppSetStream (cudaStream\_t *hStream*)**

Set the NPP CUDA stream.

**See also:**

[nppGetStream\(\)](#)

## 7.2 NPP Type Definitions and Constants

### Data Structures

- struct [NppLibraryVersion](#)
- struct [NppiPoint](#)  
*2D Point*
- struct [NppPointPolar](#)  
*2D Polar 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 [NppiHOGConfig](#)  
*The [NppiHOGConfig](#) structure defines the configuration parameters for the HOG descriptor:.*
- 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](#) ( 4294967295U )  
*Maximum 32-bit unsigned integer.*
- #define [NPP\\_MIN\\_64U](#) ( 0 )  
*Minimum 64-bit unsigned integer.*

- #define `NPP_MAX_64U` ( 18446744073709551615ULL )  
*Maximum 64-bit unsigned integer.*
- #define `NPP_MIN_8S` (-127 - 1 )  
*Minimum 8-bit signed integer.*
- #define `NPP_MAX_8S` ( 127 )  
*Maximum 8-bit signed integer.*
- #define `NPP_MIN_16S` (-32767 - 1 )  
*Minimum 16-bit signed integer.*
- #define `NPP_MAX_16S` ( 32767 )  
*Maximum 16-bit signed integer.*
- #define `NPP_MIN_32S` (-2147483647 - 1 )  
*Minimum 32-bit signed integer.*
- #define `NPP_MAX_32S` ( 2147483647 )  
*Maximum 32-bit signed integer.*
- #define `NPP_MAX_64S` ( 9223372036854775807LL )  
*Maximum 64-bit signed integer.*
- #define `NPP_MIN_64S` (-9223372036854775807LL - 1)  
*Minimum 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.*
- #define `NPP_HOG_MAX_CELL_SIZE` (16)  
*max horizontal/vertical pixel size of cell.*
- #define `NPP_HOG_MAX_BLOCK_SIZE` (64)  
*max horizontal/vertical pixel size of block.*
- #define `NPP_HOG_MAX_BINS_PER_CELL` (16)  
*max number of histogram bins.*
- #define `NPP_HOG_MAX_CELLS_PER_DESCRIPTOR` (256)

*max number of cells in a descriptor window.*

- #define `NPP_HOG_MAX_OVERLAPPING_BLOCKS_PER_DESCRIPTOR` (256)  
*max number of overlapping blocks in a descriptor window.*
- #define `NPP_HOG_MAX_DESCRIPTOR_LOCATIONS_PER_CALL` (128)  
*max number of descriptor window locations per function call.*

## Enumerations

- enum `NppiInterpolationMode` {  
`NPPI_INTER_UNDEFINED` = 0,  
`NPPI_INTER_NN` = 1,  
`NPPI_INTER_LINEAR` = 2,  
`NPPI_INTER_CUBIC` = 4,  
`NPPI_INTER_CUBIC2P_BSPLINE`,  
`NPPI_INTER_CUBIC2P_CATMULLROM`,  
`NPPI_INTER_CUBIC2P_B05C03`,  
`NPPI_INTER_SUPER` = 8,  
`NPPI_INTER_LANCZOS` = 16,  
`NPPI_INTER_LANCZOS3_ADVANCED` = 17,  
`NPPI_SMOOTH_EDGE` = (1 << 31) }  
*Filtering methods.*
- enum `NppiBayerGridPosition` {  
`NPPI_BAYER_BGGR` = 0,  
`NPPI_BAYER_RGBB` = 1,  
`NPPI_BAYER_GBRG` = 2,  
`NPPI_BAYER_GRBG` = 3 }  
*Bayer Grid Position Registration.*
- enum `NppiMaskSize` {  
`NPP_MASK_SIZE_1_X_3`,  
`NPP_MASK_SIZE_1_X_5`,  
`NPP_MASK_SIZE_3_X_1` = 100,  
`NPP_MASK_SIZE_5_X_1`,  
`NPP_MASK_SIZE_3_X_3` = 200,  
`NPP_MASK_SIZE_5_X_5`,  
`NPP_MASK_SIZE_7_X_7` = 400,  
`NPP_MASK_SIZE_9_X_9` = 500,  
`NPP_MASK_SIZE_11_X_11` = 600,  
`NPP_MASK_SIZE_13_X_13` = 700,  
`NPP_MASK_SIZE_15_X_15` = 800 }

*Fixed filter-kernel sizes.*

- enum `NppiDifferentialKernel` {  
    `NPP_FILTER_SOBEL`,  
    `NPP_FILTER_SCHARR` }

*Differential Filter types.*

- enum `NppStatus` {  
    `NPP_NOT_SUPPORTED_MODE_ERROR` = -9999,  
    `NPP_INVALID_HOST_POINTER_ERROR` = -1032,  
    `NPP_INVALID_DEVICE_POINTER_ERROR` = -1031,  
    `NPP_LUT_PALETTE_BITSIZE_ERROR` = -1030,  
    `NPP_ZC_MODE_NOT_SUPPORTED_ERROR` = -1028,  
    `NPP_NOT_SUFFICIENT_COMPUTE_CAPABILITY` = -1027,  
    `NPP_TEXTURE_BIND_ERROR` = -1024,  
    `NPP_WRONG_INTERSECTION_ROI_ERROR` = -1020,  
    `NPP_HAAR_CLASSIFIER_PIXEL_MATCH_ERROR` = -1006,  
    `NPP_MEMFREE_ERROR` = -1005,  
    `NPP_MEMSET_ERROR` = -1004,  
    `NPP_MEMCPY_ERROR` = -1003,  
    `NPP_ALIGNMENT_ERROR` = -1002,  
    `NPP_CUDA_KERNEL_EXECUTION_ERROR` = -1000,  
    `NPP_ROUND_MODE_NOT_SUPPORTED_ERROR` = -213,  
    `NPP_QUALITY_INDEX_ERROR` = -210,  
    `NPP_RESIZE_NO_OPERATION_ERROR` = -201,  
    `NPP_OVERFLOW_ERROR` = -109,  
    `NPP_NOT_EVEN_STEP_ERROR` = -108,  
    `NPP_HISTOGRAM_NUMBER_OF_LEVELS_ERROR` = -107,  
    `NPP_LUT_NUMBER_OF_LEVELS_ERROR` = -106,  
    `NPP_CORRUPTED_DATA_ERROR` = -61,  
    `NPP_CHANNEL_ORDER_ERROR` = -60,  
    `NPP_ZERO_MASK_VALUE_ERROR` = -59,  
    `NPP_QUADRANGLE_ERROR` = -58,  
    `NPP_RECTANGLE_ERROR` = -57,  
    `NPP_COEFFICIENT_ERROR` = -56,  
    `NPP_NUMBER_OF_CHANNELS_ERROR` = -53,  
    `NPP_COI_ERROR` = -52,  
    `NPP_DIVISOR_ERROR` = -51,  
    `NPP_CHANNEL_ERROR` = -47,  
    `NPP_STRIDE_ERROR` = -37,  
    `NPP_ANCHOR_ERROR` = -34,  
    `NPP_MASK_SIZE_ERROR` = -33,



```
NPP_RESIZE_FACTOR_ERROR = -23,  
NPP_INTERPOLATION_ERROR = -22,  
NPP_MIRROR_FLIP_ERROR = -21,  
NPP_MOMENT_00_ZERO_ERROR = -20,  
NPP_THRESHOLD_NEGATIVE_LEVEL_ERROR = -19,  
NPP_THRESHOLD_ERROR = -18,  
NPP_CONTEXT_MATCH_ERROR = -17,  
NPP_FFT_FLAG_ERROR = -16,  
NPP_FFT_ORDER_ERROR = -15,  
NPP_STEP_ERROR = -14,  
NPP_SCALE_RANGE_ERROR = -13,  
NPP_DATA_TYPE_ERROR = -12,  
NPP_OUT_OFF_RANGE_ERROR = -11,  
NPP_DIVIDE_BY_ZERO_ERROR = -10,  
NPP_MEMORY_ALLOCATION_ERR = -9,  
NPP_NULL_POINTER_ERROR = -8,  
NPP_RANGE_ERROR = -7,  
NPP_SIZE_ERROR = -6,  
NPP_BAD_ARGUMENT_ERROR = -5,  
NPP_NO_MEMORY_ERROR = -4,  
NPP_NOT_IMPLEMENTED_ERROR = -3,  
NPP_ERROR = -2,  
NPP_ERROR_RESERVED = -1,  
NPP_NO_ERROR = 0,  
NPP_SUCCESS = NPP_NO_ERROR,  
NPP_NO_OPERATION_WARNING = 1,  
NPP_DIVIDE_BY_ZERO_WARNING = 6,  
NPP_AFFINE_QUAD_INCORRECT_WARNING = 28,  
NPP_WRONG_INTERSECTION_ROI_WARNING = 29,  
NPP_WRONG_INTERSECTION_QUAD_WARNING = 30,  
NPP_DOUBLE_SIZE_WARNING = 35,  
NPP_MISALIGNED_DST_ROI_WARNING = 10000 }
```

*Error Status Codes.*

- `enum NppGpuComputeCapability` {  
    NPP\_CUDA\_UNKNOWN\_VERSION = -1,  
    NPP\_CUDA\_NOT\_CAPABLE = 0,  
    NPP\_CUDA\_1\_0 = 100,  
    NPP\_CUDA\_1\_1 = 110,  
    NPP\_CUDA\_1\_2 = 120,  
    NPP\_CUDA\_1\_3 = 130,  
}

```

NPP_CUDA_2_0 = 200,
NPP_CUDA_2_1 = 210,
NPP_CUDA_3_0 = 300,
NPP_CUDA_3_2 = 320,
NPP_CUDA_3_5 = 350,
NPP_CUDA_3_7 = 370,
NPP_CUDA_5_0 = 500,
NPP_CUDA_5_2 = 520,
NPP_CUDA_5_3 = 530,
NPP_CUDA_6_0 = 600,
NPP_CUDA_6_1 = 610,
NPP_CUDA_6_2 = 620,
NPP_CUDA_6_3 = 630,
NPP_CUDA_7_0 = 700 }
• 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_NEAR,
  NPP_ROUND_NEAREST_TIES_TO_EVEN = NPP_RND_NEAR,
  NPP_RND_FINANCIAL,
  NPP_ROUND_NEAREST_TIES_AWAY_FROM_ZERO = NPP_RND_FINANCIAL,
  NPP_RND_ZERO,
  NPP_ROUND_TOWARD_ZERO = NPP_RND_ZERO }
  Rounding Modes.

• enum NppiBorderType {
  NPP_BORDER_UNDEFINED = 0,
  NPP_BORDER_NONE = NPP_BORDER_UNDEFINED,
  NPP_BORDER_CONSTANT = 1,
  NPP_BORDER_REPLICATE = 2,
  NPP_BORDER_WRAP = 3,
  NPP_BORDER_MIRROR = 4 }

```

- enum `NppHintAlgorithm` {  
    `NPP_ALG_HINT_NONE`,  
    `NPP_ALG_HINT_FAST`,  
    `NPP_ALG_HINT_ACCURATE` }
- enum `NppiAlphaOp` {  
    `NPPI_OP_ALPHA_OVER`,  
    `NPPI_OP_ALPHA_IN`,  
    `NPPI_OP_ALPHA_OUT`,  
    `NPPI_OP_ALPHA_ATOP`,  
    `NPPI_OP_ALPHA_XOR`,  
    `NPPI_OP_ALPHA_PLUS`,  
    `NPPI_OP_ALPHA_OVER_PREMUL`,  
    `NPPI_OP_ALPHA_IN_PREMUL`,  
    `NPPI_OP_ALPHA_OUT_PREMUL`,  
    `NPPI_OP_ALPHA_ATOP_PREMUL`,  
    `NPPI_OP_ALPHA_XOR_PREMUL`,  
    `NPPI_OP_ALPHA_PLUS_PREMUL`,  
    `NPPI_OP_ALPHA_PREMUL` }
- enum `NppsZCType` {  
    `nppZCR`,  
    `nppZCXor`,  
    `nppZCC` }
- enum `NppiHuffmanTableType` {  
    `nppiDCTable`,  
    `nppiACTable` }
- enum `NppiNorm` {  
    `nppiNormInf = 0`,  
    `nppiNormL1 = 1`,  
    `nppiNormL2 = 2` }

### 7.2.1 Define Documentation

#### 7.2.1.1 `#define NPP_HOG_MAX_BINS_PER_CELL (16)`

max number of histogram bins.

#### 7.2.1.2 `#define NPP_HOG_MAX_BLOCK_SIZE (64)`

max horizontal/vertical pixel size of block.

#### 7.2.1.3 `#define NPP_HOG_MAX_CELL_SIZE (16)`

max horizontal/vertical pixel size of cell.

**7.2.1.4 #define NPP\_HOG\_MAX\_CELLS\_PER\_DESCRIPTOR (256)**

max number of cells in a descriptor window.

**7.2.1.5 #define NPP\_HOG\_MAX\_DESCRIPTOR\_LOCATIONS\_PER\_CALL (128)**

max number of descriptor window locations per function call.

**7.2.1.6 #define NPP\_HOG\_MAX\_OVERLAPPING\_BLOCKS\_PER\_DESCRIPTOR (256)**

max number of overlapping blocks in a descriptor window.

**7.2.1.7 #define NPP\_MAX\_16S ( 32767 )**

Maximum 16-bit signed integer.

**7.2.1.8 #define NPP\_MAX\_16U ( 65535 )**

Maximum 16-bit unsigned integer.

**7.2.1.9 #define NPP\_MAX\_32S ( 2147483647 )**

Maximum 32-bit signed integer.

**7.2.1.10 #define NPP\_MAX\_32U ( 4294967295U )**

Maximum 32-bit unsigned integer.

**7.2.1.11 #define NPP\_MAX\_64S ( 9223372036854775807LL )**

Maximum 64-bit signed integer.

**7.2.1.12 #define NPP\_MAX\_64U ( 18446744073709551615ULL )**

Maximum 64-bit unsigned integer.

**7.2.1.13 #define NPP\_MAX\_8S ( 127 )**

Maximum 8-bit signed integer.

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

Maximum 8-bit unsigned integer.

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

Largest positive 32-bit floating point value.

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

Largest positive 64-bit floating point value.

**7.2.1.17 #define NPP\_MIN\_16S (-32767 - 1 )**

Minimum 16-bit signed integer.

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

Minimum 16-bit unsigned integer.

**7.2.1.19 #define NPP\_MIN\_32S (-2147483647 - 1 )**

Minimum 32-bit signed integer.

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

Minimum 32-bit unsigned integer.

**7.2.1.21 #define NPP\_MIN\_64S (-9223372036854775807LL - 1 )**

Minimum 64-bit signed integer.

**7.2.1.22 #define NPP\_MIN\_64U ( 0 )**

Minimum 64-bit unsigned integer.

**7.2.1.23 #define NPP\_MIN\_8S (-127 - 1 )**

Minimum 8-bit signed integer.

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

Minimum 8-bit unsigned integer.

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

Smallest positive 32-bit floating point value.

### 7.2.1.26 #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.  
*NPP\_CUDA\_1\_0* Indicates that CUDA 1.0 capable device is machine's default device.  
*NPP\_CUDA\_1\_1* Indicates that CUDA 1.1 capable device is machine's default device.  
*NPP\_CUDA\_1\_2* Indicates that CUDA 1.2 capable device is machine's default device.  
*NPP\_CUDA\_1\_3* Indicates that CUDA 1.3 capable device is machine's default device.  
*NPP\_CUDA\_2\_0* Indicates that CUDA 2.0 capable device is machine's default device.  
*NPP\_CUDA\_2\_1* Indicates that CUDA 2.1 capable device is machine's default device.  
*NPP\_CUDA\_3\_0* Indicates that CUDA 3.0 capable device is machine's default device.  
*NPP\_CUDA\_3\_2* Indicates that CUDA 3.2 capable device is machine's default device.  
*NPP\_CUDA\_3\_5* Indicates that CUDA 3.5 capable device is machine's default device.  
*NPP\_CUDA\_3\_7* Indicates that CUDA 3.7 capable device is machine's default device.  
*NPP\_CUDA\_5\_0* Indicates that CUDA 5.0 capable device is machine's default device.  
*NPP\_CUDA\_5\_2* Indicates that CUDA 5.2 capable device is machine's default device.  
*NPP\_CUDA\_5\_3* Indicates that CUDA 5.3 capable device is machine's default device.  
*NPP\_CUDA\_6\_0* Indicates that CUDA 6.0 capable device is machine's default device.  
*NPP\_CUDA\_6\_1* Indicates that CUDA 6.1 capable device is machine's default device.  
*NPP\_CUDA\_6\_2* Indicates that CUDA 6.2 capable device is machine's default device.  
*NPP\_CUDA\_6\_3* Indicates that CUDA 6.3 capable device is machine's default device.  
*NPP\_CUDA\_7\_0* Indicates that CUDA 7.0 or better is machine's default device.

### 7.2.2.3 enum NppHintAlgorithm

Enumerator:

*NPP\_ALG\_HINT\_NONE*  
*NPP\_ALG\_HINT\_FAST*  
*NPP\_ALG\_HINT\_ACCURATE*

### 7.2.2.4 enum NppiAlphaOp

Enumerator:

*NPPI\_OP\_ALPHA\_OVER*  
*NPPI\_OP\_ALPHA\_IN*  
*NPPI\_OP\_ALPHA\_OUT*  
*NPPI\_OP\_ALPHA\_ATOP*  
*NPPI\_OP\_ALPHA\_XOR*  
*NPPI\_OP\_ALPHA\_PLUS*  
*NPPI\_OP\_ALPHA\_OVER\_PREMUL*  
*NPPI\_OP\_ALPHA\_IN\_PREMUL*  
*NPPI\_OP\_ALPHA\_OUT\_PREMUL*  
*NPPI\_OP\_ALPHA\_ATOP\_PREMUL*  
*NPPI\_OP\_ALPHA\_XOR\_PREMUL*  
*NPPI\_OP\_ALPHA\_PLUS\_PREMUL*  
*NPPI\_OP\_ALPHA\_PREMUL*

### 7.2.2.5 enum NppiAxis

Enumerator:

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

### 7.2.2.6 enum NppiBayerGridPosition

Bayer Grid Position Registration.

Enumerator:

*NPPI\_BAYER\_BGGR* Default registration position.  
*NPPI\_BAYER\_RGGB*  
*NPPI\_BAYER\_GBRG*  
*NPPI\_BAYER\_GRBG*

### 7.2.2.7 enum NppiBorderType

Enumerator:

*NPP\_BORDER\_UNDEFINED*  
*NPP\_BORDER\_NONE*  
*NPP\_BORDER\_CONSTANT*  
*NPP\_BORDER\_REPLICATE*  
*NPP\_BORDER\_WRAP*  
*NPP\_BORDER\_MIRROR*

### 7.2.2.8 enum NppiDifferentialKernel

Differential Filter types.

Enumerator:

*NPP\_FILTER\_SOBEL*  
*NPP\_FILTER\_SCHARR*

### 7.2.2.9 enum NppiHuffmanTableType

Enumerator:

*nppiDCTable* DC Table.  
*nppiACTable* AC Table.

### 7.2.2.10 enum NppiInterpolationMode

Filtering methods.

Enumerator:

*NPPI\_INTER\_UNDEFINED*  
*NPPI\_INTER\_NN* Nearest neighbor filtering.  
*NPPI\_INTER\_LINEAR* Linear interpolation.  
*NPPI\_INTER\_CUBIC* Cubic interpolation.  
*NPPI\_INTER\_CUBIC2P\_BSPLINE* Two-parameter cubic filter (B=1, C=0).  
*NPPI\_INTER\_CUBIC2P\_CATMULLROM* Two-parameter cubic filter (B=0, C=1/2).  
*NPPI\_INTER\_CUBIC2P\_B05C03* Two-parameter cubic filter (B=1/2, C=3/10).  
*NPPI\_INTER\_SUPER* Super sampling.  
*NPPI\_INTER\_LANCZOS* Lanczos filtering.  
*NPPI\_INTER\_LANCZOS3\_ADVANCED* Generic Lanczos filtering with order 3.  
*NPPI\_SMOOTH\_EDGE* Smooth edge filtering.



**7.2.2.11 enum NppiMaskSize**

Fixed filter-kernel sizes.

**Enumerator:**

*NPP\_MASK\_SIZE\_1\_X\_3*  
*NPP\_MASK\_SIZE\_1\_X\_5*  
*NPP\_MASK\_SIZE\_3\_X\_1*  
*NPP\_MASK\_SIZE\_5\_X\_1*  
*NPP\_MASK\_SIZE\_3\_X\_3*  
*NPP\_MASK\_SIZE\_5\_X\_5*  
*NPP\_MASK\_SIZE\_7\_X\_7*  
*NPP\_MASK\_SIZE\_9\_X\_9*  
*NPP\_MASK\_SIZE\_11\_X\_11*  
*NPP\_MASK\_SIZE\_13\_X\_13*  
*NPP\_MASK\_SIZE\_15\_X\_15*

**7.2.2.12 enum NppiNorm****Enumerator:**

*nppiNormInf* maximum  
*nppiNormL1* sum  
*nppiNormL2* square root of sum of squares

**7.2.2.13 enum NppRoundMode**

Rounding Modes.

The enumerated rounding modes are used by a large number of NPP primitives to allow the user to specify the method by which fractional values are converted to integer values. Also see [Rounding Modes](#).

For NPP release 5.5 new names for the three rounding modes are introduced that are based on the naming conventions for rounding modes set forth in the IEEE-754 floating-point standard. Developers are encouraged to use the new, longer names to be future proof as the legacy names will be deprecated in subsequent NPP releases.

**Enumerator:**

*NPP\_RND\_NEAR* Round to the nearest even integer.  
 All fractional numbers are rounded to their nearest integer. The ambiguous cases (i.e.  $\langle \text{integer} \rangle.5$ ) are rounded to the closest even integer. E.g.

- $\text{roundNear}(0.5) = 0$
- $\text{roundNear}(0.6) = 1$
- $\text{roundNear}(1.5) = 2$
- $\text{roundNear}(-1.5) = -2$

*NPP\_ROUND\_NEAREST\_TIES\_TO\_EVEN* Alias name for *NPP\_RND\_NEAR*.

***NPP\_RND\_FINANCIAL*** Round according to financial rule.

All fractional numbers are rounded to their nearest integer. The ambiguous cases (i.e.  $\langle \text{integer} \rangle .5$ ) are rounded away from zero. E.g.

- `roundFinancial(0.4) = 0`
- `roundFinancial(0.5) = 1`
- `roundFinancial(-1.5) = -2`

***NPP\_ROUND\_NEAREST\_TIES\_AWAY\_FROM\_ZERO*** Alias name for [NPP\\_RND\\_FINANCIAL](#).

***NPP\_RND\_ZERO*** Round towards zero (truncation).

All fractional numbers of the form  $\langle \text{integer} \rangle . \langle \text{decimals} \rangle$  are truncated to  $\langle \text{integer} \rangle$ .

- `roundZero(1.5) = 1`
- `roundZero(1.9) = 1`
- `roundZero(-2.5) = -2`

***NPP\_ROUND\_TOWARD\_ZERO*** Alias name for [NPP\\_RND\\_ZERO](#).

#### 7.2.2.14 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\_INVALID\_HOST\_POINTER\_ERROR***

***NPP\_INVALID\_DEVICE\_POINTER\_ERROR***

***NPP\_LUT\_PALETTE\_BITSIZE\_ERROR***

***NPP\_ZC\_MODE\_NOT\_SUPPORTED\_ERROR*** ZeroCrossing mode not supported.

***NPP\_NOT\_SUFFICIENT\_COMPUTE\_CAPABILITY***

***NPP\_TEXTURE\_BIND\_ERROR***

***NPP\_WRONG\_INTERSECTION\_ROI\_ERROR***

***NPP\_HAAR\_CLASSIFIER\_PIXEL\_MATCH\_ERROR***

***NPP\_MEMFREE\_ERROR***

***NPP\_MEMSET\_ERROR***

***NPP\_MEMCPY\_ERROR***

***NPP\_ALIGNMENT\_ERROR***

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

***NPP\_ROUND\_MODE\_NOT\_SUPPORTED\_ERROR*** Unsupported round mode.

***NPP\_QUALITY\_INDEX\_ERROR*** Image pixels are constant for quality index.

***NPP\_RESIZE\_NO\_OPERATION\_ERROR*** One of the output image dimensions is less than 1 pixel.

***NPP\_OVERFLOW\_ERROR*** Number overflows the upper or lower limit of the data type.

***NPP\_NOT\_EVEN\_STEP\_ERROR*** Step value is not pixel multiple.

***NPP\_HISTOGRAM\_NUMBER\_OF\_LEVELS\_ERROR*** Number of levels for histogram is less than 2.

***NPP\_LUT\_NUMBER\_OF\_LEVELS\_ERROR*** Number of levels for LUT is less than 2.

***NPP\_CORRUPTED\_DATA\_ERROR*** Processed data is corrupted.

***NPP\_CHANNEL\_ORDER\_ERROR*** Wrong order of the destination channels.

***NPP\_ZERO\_MASK\_VALUE\_ERROR*** All values of the mask are zero.

***NPP\_QUADRANGLE\_ERROR*** The quadrangle is nonconvex or degenerates into triangle, line or point.

***NPP\_RECTANGLE\_ERROR*** Size of the rectangle region is less than or equal to 1.

***NPP\_COEFFICIENT\_ERROR*** Unallowable values of the transformation coefficients.

***NPP\_NUMBER\_OF\_CHANNELS\_ERROR*** Bad or unsupported number of channels.

***NPP\_COI\_ERROR*** Channel of interest is not 1, 2, or 3.

***NPP\_DIVISOR\_ERROR*** Divisor is equal to zero.

***NPP\_CHANNEL\_ERROR*** Illegal channel index.

***NPP\_STRIDE\_ERROR*** Stride is less than the row length.

***NPP\_ANCHOR\_ERROR*** Anchor point is outside mask.

***NPP\_MASK\_SIZE\_ERROR*** Lower bound is larger than upper bound.

***NPP\_RESIZE\_FACTOR\_ERROR***

***NPP\_INTERPOLATION\_ERROR***

***NPP\_MIRROR\_FLIP\_ERROR***

***NPP\_MOMENT\_00\_ZERO\_ERROR***

***NPP\_THRESHOLD\_NEGATIVE\_LEVEL\_ERROR***

***NPP\_THRESHOLD\_ERROR***

***NPP\_CONTEXT\_MATCH\_ERROR***

***NPP\_FFT\_FLAG\_ERROR***

***NPP\_FFT\_ORDER\_ERROR***

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

***NPP\_SCALE\_RANGE\_ERROR***

***NPP\_DATA\_TYPE\_ERROR***

***NPP\_OUT\_OFF\_RANGE\_ERROR***

***NPP\_DIVIDE\_BY\_ZERO\_ERROR***

***NPP\_MEMORY\_ALLOCATION\_ERR***

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

***NPP\_RANGE\_ERROR***

***NPP\_SIZE\_ERROR***

***NPP\_BAD\_ARGUMENT\_ERROR***

***NPP\_NO\_MEMORY\_ERROR***

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

***NPP\_ERROR***

***NPP\_ERROR\_RESERVED***

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

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

***NPP\_NO\_OPERATION\_WARNING*** Indicates that no operation was performed.

***NPP\_DIVIDE\_BY\_ZERO\_WARNING*** Divisor is zero however does not terminate the execution.

***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\_WRONG\_INTERSECTION\_ROI\_WARNING*** The given ROI has no intersection with either the source or destination ROI.

Thus no operation was performed.

***NPP\_WRONG\_INTERSECTION\_QUAD\_WARNING*** The given quadrangle has no intersection with either the source or destination ROI.

Thus no operation was performed.

***NPP\_DOUBLE\_SIZE\_WARNING*** Image size isn't multiple of two.

Indicates that in case of 422/411/420 sampling the ROI width/height was modified for proper processing.

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

#### 7.2.2.15 enum NppsZCType

##### Enumerator:

***nppZCR*** sign change

***nppZCXor*** sign change XOR

***nppZCC*** sign change count\_0

## 7.3 Basic NPP Data Types

### Data Structures

- struct [NPP\\_ALIGN\\_8](#)  
*Complex Number This struct represents an unsigned int complex number.*
- struct [NPP\\_ALIGN\\_16](#)  
*Complex Number This struct represents a long long 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*
- typedef struct [NPP\\_ALIGN\\_8](#) [Npp32uc](#)  
*Complex Number This struct represents an unsigned int complex number.*
- typedef struct [NPP\\_ALIGN\\_8](#) [Npp32sc](#)  
*Complex Number This struct represents a signed int complex number.*

- typedef struct [NPP\\_ALIGN\\_8 Npp32fc](#)  
*Complex Number This struct represents a single floating-point complex number.*
- typedef struct [NPP\\_ALIGN\\_16 Npp64sc](#)  
*Complex Number This struct represents a long long complex number.*
- typedef struct [NPP\\_ALIGN\\_16 Npp64fc](#)  
*Complex Number This struct represents a double floating-point complex number.*

## Functions

- struct [\\_\\_align\\_\\_](#) (2)  
*Complex Number This struct represents an unsigned char complex number.*
- struct [\\_\\_align\\_\\_](#) (4)  
*Complex Number This struct represents an unsigned short complex number.*

## Variables

- [Npp8uc](#)
- [Npp16uc](#)
- [Npp16sc](#)

### 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 struct NPP\_ALIGN\_8 Npp32fc

Complex Number This struct represents a single floating-point complex number.

#### 7.3.1.5 typedef int Npp32s

32-bit signed integers

**7.3.1.6 typedef struct NPP\_ALIGN\_8 Npp32sc**

Complex Number This struct represents a signed int complex number.

**7.3.1.7 typedef unsigned int Npp32u**

32-bit unsigned integers

**7.3.1.8 typedef struct NPP\_ALIGN\_8 Npp32uc**

Complex Number This struct represents an unsigned int complex number.

**7.3.1.9 typedef double Npp64f**

64-bit floating-point numbers

**7.3.1.10 typedef struct NPP\_ALIGN\_16 Npp64fc**

Complex Number This struct represents a double floating-point complex number.

**7.3.1.11 typedef long long Npp64s**

64-bit signed integers

**7.3.1.12 typedef struct NPP\_ALIGN\_16 Npp64sc**

Complex Number This struct represents a long long complex number.

**7.3.1.13 typedef unsigned long long Npp64u**

64-bit unsigned integers

**7.3.1.14 typedef signed char Npp8s**

8-bit signed chars

**7.3.1.15 typedef unsigned char Npp8u**

8-bit unsigned chars

**7.3.2 Function Documentation****7.3.2.1 struct \_\_align\_\_ (4) [read]**

Complex Number This struct represents an unsigned short complex number.

Complex Number This struct represents a short complex number.

< Real part

< Imaginary part

< Real part

< Imaginary part

### **7.3.2.2 struct \_\_align\_\_ (2) [read]**

Complex Number This struct represents an unsigned char complex number.

< Real part

< Imaginary part

## **7.3.3 Variable Documentation**

### **7.3.3.1 Npp16sc**

### **7.3.3.2 Npp16uc**

### **7.3.3.3 Npp8uc**



## 7.4 Threshold and Compare Operations

Methods for pixel-wise threshold and compare operations.

### Modules

- [Threshold Operations](#)

*Threshold image pixels.*

- [Compare Operations](#)

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

### 7.4.1 Detailed Description

Methods for pixel-wise threshold and compare operations.

These functions can be found in the nppitc library. Linking to only the sub-libraries that you use can significantly save link time, application load time, and CUDA runtime startup time when using dynamic libraries.

## 7.5 Threshold Operations

Threshold image pixels.

### Functions

- `NppStatus nppiThreshold_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_8u_C1IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, `NppCmpOp` eComparisonOperation)  
*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], `NppCmpOp` eComparisonOperation)  
*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], `NppCmpOp` eComparisonOperation)  
*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit unsigned short threshold.*

- `NppStatus nppiThreshold_16u_C3IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_16s_C3R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_16s_C3IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_32f_C3R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_32f_C3IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_8u_AC4R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_8u_AC4IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_16u_AC4R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_16u_AC4IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_16s_AC4R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_16s_AC4IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_32f_AC4R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 32-bit floating point image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_32f_AC4IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_GT_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_GT_8u_C1IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold)  
*1 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_GT_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold)  
*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_GT_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold)  
*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_GT_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold)  
*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_GT_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold)  
*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_GT_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold)  
*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_GT_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold)  
*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_GT_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3])  
*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_GT_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3])  
*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_GT_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3])  
*3 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_GT_16u_C3IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3])  
*3 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_GT_16s_C3R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3])

*3 channel 16-bit signed short threshold.*

- `NppStatus nppiThreshold_GT_16s_C3IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`)

*3 channel 16-bit signed short in place threshold.*

- `NppStatus nppiThreshold_GT_32f_C3R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`)

*3 channel 32-bit floating point threshold.*

- `NppStatus nppiThreshold_GT_32f_C3IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`)

*3 channel 32-bit floating point in place threshold.*

- `NppStatus nppiThreshold_GT_8u_AC4R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`)

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

- `NppStatus nppiThreshold_GT_8u_AC4IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`)

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

- `NppStatus nppiThreshold_GT_16u_AC4R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`)

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

- `NppStatus nppiThreshold_GT_16u_AC4IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`)

*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_GT_16s_AC4R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`)

*4 channel 16-bit signed short image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_GT_16s_AC4IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`)

*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_GT_32f_AC4R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`)

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

- `NppStatus nppiThreshold_GT_32f_AC4IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`)

*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_LT_8u_C1R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u nThreshold`)

*1 channel 8-bit unsigned char threshold.*

- `NppStatus nppiThreshold_LT_8u_C1R` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u nThreshold`)  
*1 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_LT_16u_C1R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u nThreshold`)  
*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_LT_16u_C1IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u nThreshold`)  
*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_LT_16s_C1R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s nThreshold`)  
*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_LT_16s_C1IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s nThreshold`)  
*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_LT_32f_C1R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f nThreshold`)  
*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_LT_32f_C1IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f nThreshold`)  
*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_LT_8u_C3R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`)  
*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_LT_8u_C3IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`)  
*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_LT_16u_C3R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`)  
*3 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_LT_16u_C3IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`)  
*3 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_LT_16s_C3R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`)  
*3 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_LT_16s_C3IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`)  
*3 channel 16-bit signed short in place threshold.*

- `NppStatus nppiThreshold_LT_32f_C3R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3])  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_LT_32f_C3IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3])  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_LT_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3])  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_8u_AC4IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3])  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_16u_AC4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3])  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_16u_AC4IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3])  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_16s_AC4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3])  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_16s_AC4IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3])  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_32f_AC4R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3])  
*4 channel 32-bit floating point image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LT_32f_AC4IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3])  
*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, const `Npp8u` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_Val_8u_C1IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, const `Npp8u` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 8-bit unsigned char in place threshold.*

- `NppStatus nppiThreshold_Val_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_Val_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_Val_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_Val_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_Val_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_Val_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue, `NppCmpOp` eComparisonOperation)  
*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_Val_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3], `NppCmpOp` eComparisonOperation)  
*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_Val_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3], `NppCmpOp` eComparisonOperation)  
*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_Val_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3], `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_Val_16u_C3IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3], `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_Val_16s_C3R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3], `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit signed short threshold.*



- `NppStatus nppiThreshold_Val_16s_C3IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_Val_32f_C3R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_Val_32f_C3IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`, `NppCmpOp eComparisonOperation`)  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_Val_8u_AC4R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `const Npp8u rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_8u_AC4IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `const Npp8u rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_16u_AC4R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `const Npp16u rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_16u_AC4IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `const Npp16u rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_16s_AC4R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_16s_AC4IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_32f_AC4R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 32-bit floating point image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_Val_32f_AC4IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`, `NppCmpOp eComparisonOperation`)  
*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_GTVal_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, const `Npp8u` nValue)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_GTVal_8u_C1IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` nThreshold, const `Npp8u` nValue)  
*1 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_GTVal_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue)  
*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_GTVal_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue)  
*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_GTVal_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue)  
*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_GTVal_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue)  
*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_GTVal_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue)  
*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_GTVal_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue)  
*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_GTVal_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])  
*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_GTVal_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])  
*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_GTVal_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])  
*3 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_GTVal_16u_C3IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])  
*3 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_GTVal_16s_C3R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3])  
*3 channel 16-bit signed short threshold.*

- `NppStatus nppiThreshold_GTVal_16s_C3IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`)  
*3 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_GTVal_32f_C3R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`)  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_GTVal_32f_C3IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`)  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_GTVal_8u_AC4R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `const Npp8u rValues[3]`)  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_8u_AC4IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u rThresholds[3]`, `const Npp8u rValues[3]`)  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_16u_AC4R` (`const Npp16u *pSrc`, `int nSrcStep`, `Npp16u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `const Npp16u rValues[3]`)  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_16u_AC4IR` (`Npp16u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16u rThresholds[3]`, `const Npp16u rValues[3]`)  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_16s_AC4R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`)  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_16s_AC4IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `const Npp16s rValues[3]`)  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_32f_AC4R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`)  
*4 channel 32-bit floating point image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_GTVal_32f_AC4IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholds[3]`, `const Npp32f rValues[3]`)  
*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_8u_C1R` (`const Npp8u *pSrc`, `int nSrcStep`, `Npp8u *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp8u nThreshold`, `const Npp8u nValue`)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_LTVal_8u_C1IR` (`Npp8u *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp8u nThreshold`, `const Npp8u nValue`)

*1 channel 8-bit unsigned char in place threshold.*

- `NppStatus nppiThreshold_LTVal_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue)

*1 channel 16-bit unsigned short threshold.*

- `NppStatus nppiThreshold_LTVal_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThreshold, const `Npp16u` nValue)

*1 channel 16-bit unsigned short in place threshold.*

- `NppStatus nppiThreshold_LTVal_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue)

*1 channel 16-bit signed short threshold.*

- `NppStatus nppiThreshold_LTVal_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThreshold, const `Npp16s` nValue)

*1 channel 16-bit signed short in place threshold.*

- `NppStatus nppiThreshold_LTVal_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue)

*1 channel 32-bit floating point threshold.*

- `NppStatus nppiThreshold_LTVal_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThreshold, const `Npp32f` nValue)

*1 channel 32-bit floating point in place threshold.*

- `NppStatus nppiThreshold_LTVal_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])

*3 channel 8-bit unsigned char threshold.*

- `NppStatus nppiThreshold_LTVal_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])

*3 channel 8-bit unsigned char in place threshold.*

- `NppStatus nppiThreshold_LTVal_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])

*3 channel 16-bit unsigned short threshold.*

- `NppStatus nppiThreshold_LTVal_16u_C3IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])

*3 channel 16-bit unsigned short in place threshold.*

- `NppStatus nppiThreshold_LTVal_16s_C3R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3])

*3 channel 16-bit signed short threshold.*

- `NppStatus nppiThreshold_LTVal_16s_C3IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3])

*3 channel 16-bit signed short in place threshold.*

- `NppStatus nppiThreshold_LTVal_32f_C3R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3], const `Npp32f` rValues[3])  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_LTVal_32f_C3IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3], const `Npp32f` rValues[3])  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_LTVal_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_8u_AC4IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholds[3], const `Npp8u` rValues[3])  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_16u_AC4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_16u_AC4IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholds[3], const `Npp16u` rValues[3])  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_16s_AC4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3])  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_16s_AC4IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholds[3], const `Npp16s` rValues[3])  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_32f_AC4R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3], const `Npp32f` rValues[3])  
*4 channel 32-bit floating point image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTVal_32f_AC4IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholds[3], const `Npp32f` rValues[3])  
*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_8u_C1R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` nThresholdLT, const `Npp8u` nValueLT, const `Npp8u` nThresholdGT, const `Npp8u` nValueGT)  
*1 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_LTValGTVal_8u_C1IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` nThresholdLT, const `Npp8u` nValueLT, const `Npp8u` nThresholdGT, const `Npp8u` nValueGT)  
*1 channel 8-bit unsigned char in place threshold.*

- `NppStatus nppiThreshold_LTValGTVal_16u_C1R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` nThresholdLT, const `Npp16u` nValueLT, const `Npp16u` nThresholdGT, const `Npp16u` nValueGT)

*1 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16u_C1IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` nThresholdLT, const `Npp16u` nValueLT, const `Npp16u` nThresholdGT, const `Npp16u` nValueGT)

*1 channel 16-bit unsigned short in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16s_C1R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` nThresholdLT, const `Npp16s` nValueLT, const `Npp16s` nThresholdGT, const `Npp16s` nValueGT)

*1 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16s_C1IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` nThresholdLT, const `Npp16s` nValueLT, const `Npp16s` nThresholdGT, const `Npp16s` nValueGT)

*1 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_32f_C1R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` nThresholdLT, const `Npp32f` nValueLT, const `Npp32f` nThresholdGT, const `Npp32f` nValueGT)

*1 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_LTValGTVal_32f_C1IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` nThresholdLT, const `Npp32f` nValueLT, const `Npp32f` nThresholdGT, const `Npp32f` nValueGT)

*1 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_8u_C3R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholdsLT[3], const `Npp8u` rValuesLT[3], const `Npp8u` rThresholdsGT[3], const `Npp8u` rValuesGT[3])

*3 channel 8-bit unsigned char threshold.*
- `NppStatus nppiThreshold_LTValGTVal_8u_C3IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholdsLT[3], const `Npp8u` rValuesLT[3], const `Npp8u` rThresholdsGT[3], const `Npp8u` rValuesGT[3])

*3 channel 8-bit unsigned char in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16u_C3R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholdsLT[3], const `Npp16u` rValuesLT[3], const `Npp16u` rThresholdsGT[3], const `Npp16u` rValuesGT[3])

*3 channel 16-bit unsigned short threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16u_C3IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholdsLT[3], const `Npp16u` rValuesLT[3], const `Npp16u` rThresholdsGT[3], const `Npp16u` rValuesGT[3])

*3 channel 16-bit unsigned short in place threshold.*



- `NppStatus nppiThreshold_LTValGTVal_16s_C3R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholdsLT[3], const `Npp16s` rValuesLT[3], const `Npp16s` rThresholdsGT[3], const `Npp16s` rValuesGT[3])  
*3 channel 16-bit signed short threshold.*
- `NppStatus nppiThreshold_LTValGTVal_16s_C3IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholdsLT[3], const `Npp16s` rValuesLT[3], const `Npp16s` rThresholdsGT[3], const `Npp16s` rValuesGT[3])  
*3 channel 16-bit signed short in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_32f_C3R` (const `Npp32f` \*pSrc, int nSrcStep, `Npp32f` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholdsLT[3], const `Npp32f` rValuesLT[3], const `Npp32f` rThresholdsGT[3], const `Npp32f` rValuesGT[3])  
*3 channel 32-bit floating point threshold.*
- `NppStatus nppiThreshold_LTValGTVal_32f_C3IR` (`Npp32f` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp32f` rThresholdsLT[3], const `Npp32f` rValuesLT[3], const `Npp32f` rThresholdsGT[3], const `Npp32f` rValuesGT[3])  
*3 channel 32-bit floating point in place threshold.*
- `NppStatus nppiThreshold_LTValGTVal_8u_AC4R` (const `Npp8u` \*pSrc, int nSrcStep, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholdsLT[3], const `Npp8u` rValuesLT[3], const `Npp8u` rThresholdsGT[3], const `Npp8u` rValuesGT[3])  
*4 channel 8-bit unsigned char image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_8u_AC4IR` (`Npp8u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp8u` rThresholdsLT[3], const `Npp8u` rValuesLT[3], const `Npp8u` rThresholdsGT[3], const `Npp8u` rValuesGT[3])  
*4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_16u_AC4R` (const `Npp16u` \*pSrc, int nSrcStep, `Npp16u` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholdsLT[3], const `Npp16u` rValuesLT[3], const `Npp16u` rThresholdsGT[3], const `Npp16u` rValuesGT[3])  
*4 channel 16-bit unsigned short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_16u_AC4IR` (`Npp16u` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16u` rThresholdsLT[3], const `Npp16u` rValuesLT[3], const `Npp16u` rThresholdsGT[3], const `Npp16u` rValuesGT[3])  
*4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_16s_AC4R` (const `Npp16s` \*pSrc, int nSrcStep, `Npp16s` \*pDst, int nDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholdsLT[3], const `Npp16s` rValuesLT[3], const `Npp16s` rThresholdsGT[3], const `Npp16s` rValuesGT[3])  
*4 channel 16-bit signed short image threshold, not affecting Alpha.*
- `NppStatus nppiThreshold_LTValGTVal_16s_AC4IR` (`Npp16s` \*pSrcDst, int nSrcDstStep, `NppiSize` oSizeROI, const `Npp16s` rThresholdsLT[3], const `Npp16s` rValuesLT[3], const `Npp16s` rThresholdsGT[3], const `Npp16s` rValuesGT[3])  
*4 channel 16-bit signed short in place image threshold, not affecting Alpha.*

- `NppStatus nppiThreshold_LTValGTVal_32f_AC4R` (`const Npp32f *pSrc`, `int nSrcStep`, `Npp32f *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholdsLT[3]`, `const Npp32f rValuesLT[3]`, `const Npp32f rThresholdsGT[3]`, `const Npp32f rValuesGT[3]`)

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

- `NppStatus nppiThreshold_LTValGTVal_32f_AC4IR` (`Npp32f *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp32f rThresholdsLT[3]`, `const Npp32f rValuesLT[3]`, `const Npp32f rThresholdsGT[3]`, `const Npp32f rValuesGT[3]`)

*4 channel 32-bit floating point in place image threshold, not affecting Alpha.*

## 7.5.1 Detailed Description

Threshold image pixels.

## 7.5.2 Function Documentation

### 7.5.2.1 `NppStatus nppiThreshold_16s_AC4IR` (`Npp16s *pSrcDst`, `int nSrcDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)

4 channel 16-bit signed short in place 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:

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

### 7.5.2.2 `NppStatus nppiThreshold_16s_AC4R` (`const Npp16s *pSrc`, `int nSrcStep`, `Npp16s *pDst`, `int nDstStep`, `NppiSize oSizeROI`, `const Npp16s rThresholds[3]`, `NppCmpOp eComparisonOperation`)

4 channel 16-bit signed short 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).

*rThresholds* 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.

**7.5.2.3 NppStatus nppiThreshold\_16s\_C1IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold, NppCmpOp eComparisonOperation)**

1 channel 16-bit signed short in place 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:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-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.

**7.5.2.4 NppStatus nppiThreshold\_16s\_C1R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold, NppCmpOp eComparisonOperation)**

1 channel 16-bit signed short 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.

**7.5.2.5 NppStatus nppiThreshold\_16s\_C3IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 16-bit signed short in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

**7.5.2.6 NppStatus nppiThreshold\_16s\_C3R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 16-bit signed short 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 \(ROI\)](#).

*rThresholds* 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.

#### 7.5.2.7 `NppStatus nppiThreshold_16u_AC4IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit unsigned short in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

#### 7.5.2.8 `NppStatus nppiThreshold_16u_AC4R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit unsigned short 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 \(ROI\)](#).

*rThresholds* 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.

### 7.5.2.9 `NppStatus nppiThreshold_16u_C1IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold, NppCmpOp eComparisonOperation)`

1 channel 16-bit unsigned short in place 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:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-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.

### 7.5.2.10 `NppStatus nppiThreshold_16u_C1R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold, NppCmpOp eComparisonOperation)`

1 channel 16-bit unsigned short 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.

### 7.5.2.11 `NppStatus nppiThreshold_16u_C3IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], NppCmpOp eComparisonOperation)`

3 channel 16-bit unsigned short in place 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:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* 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.

#### 7.5.2.12 NppStatus nppiThreshold\_16u\_C3R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], NppCmpOp eComparisonOperation)

3 channel 16-bit unsigned short 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).

*rThresholds* 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.

#### 7.5.2.13 NppStatus nppiThreshold\_32f\_AC4IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], NppCmpOp eComparisonOperation)

4 channel 32-bit floating point in place 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:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

**7.5.2.14 NppStatus nppiThreshold\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], NppCmpOp eComparisonOperation)**

4 channel 32-bit floating point 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 \(ROI\)](#).

*rThresholds* 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.

**7.5.2.15 NppStatus nppiThreshold\_32f\_C1IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold, NppCmpOp eComparisonOperation)**

1 channel 32-bit floating point in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-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.

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

1 channel 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.

**7.5.2.17 NppStatus nppiThreshold\_32f\_C3IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 32-bit floating point in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

**7.5.2.18 NppStatus nppiThreshold\_32f\_C3R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 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 \(ROI\)](#).

*rThresholds* 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.

**7.5.2.19 NppStatus nppiThreshold\_8u\_AC4IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], NppCmpOp eComparisonOperation)**

4 channel 8-bit unsigned char in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.



**7.5.2.20** `NppStatus nppiThreshold_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned char 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).

*rThresholds* 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.

**7.5.2.21** `NppStatus nppiThreshold_8u_C1IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold, NppCmpOp eComparisonOperation)`

1 channel 8-bit unsigned char in place 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:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-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.

**7.5.2.22 NppStatus nppiThreshold\_8u\_C1R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold, NppCmpOp eComparisonOperation)**

1 channel 8-bit unsigned char 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.

**7.5.2.23 NppStatus nppiThreshold\_8u\_C3IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 8-bit unsigned char in place 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:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* 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.

**7.5.2.24 NppStatus nppiThreshold\_8u\_C3R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], NppCmpOp eComparisonOperation)**

3 channel 8-bit unsigned char 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).

*rThresholds* 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.

**7.5.2.25 NppStatus nppiThreshold\_GT\_16s\_AC4IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])**

4 channel 16-bit signed short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.26 NppStatus nppiThreshold\_GT\_16s\_AC4R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])**

4 channel 16-bit signed short image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.27 **NppStatus nppiThreshold\_GT\_16s\_C1IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold)**

1 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.28 **NppStatus nppiThreshold\_GT\_16s\_C1R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold)**

1 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is greater than 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.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

### 7.5.2.29 **NppStatus nppiThreshold\_GT\_16s\_C3IR** (Npp16s \* *pSrcDst*, int *nSrcDstStep*, NppiSize *oSizeROI*, const Npp16s *rThresholds*[3])

3 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.30 **NppStatus nppiThreshold\_GT\_16s\_C3R** (const Npp16s \* *pSrc*, int *nSrcStep*, Npp16s \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp16s *rThresholds*[3])

3 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is greater than 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).

*rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.31 **NppStatus nppiThreshold\_GT\_16u\_AC4IR** (Npp16u \* *pSrcDst*, int *nSrcDstStep*, NppiSize *oSizeROI*, const Npp16u *rThresholds*[3])

4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.32 NppStatus nppiThreshold\_GT\_16u\_AC4R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])**

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

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set 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).

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.33 NppStatus nppiThreshold\_GT\_16u\_C1IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold)**

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.34 NppStatus nppiThreshold\_GT\_16u\_C1R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold)**

1 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is greater than 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.

**Returns:**

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

**7.5.2.35 NppStatus nppiThreshold\_GT\_16u\_C3IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])**

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

- pSrcDst* In-Place Image Pointer.
- nSrcDstStep* In-Place-Image Line Step.
- oSizeROI* Region-of-Interest (ROI).
- rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.36 NppStatus nppiThreshold\_GT\_16u\_C3R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])**

3 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is greater than 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.37 **NppStatus nppiThreshold\_GT\_32f\_AC4IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])**

4 channel 32-bit floating point in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.38 **NppStatus nppiThreshold\_GT\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])**

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

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.



### 7.5.2.39 `NppStatus nppiThreshold_GT_32f_C1IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold)`

1 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes.

### 7.5.2.40 `NppStatus nppiThreshold_GT_32f_C1R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f nThreshold)`

1 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is greater than 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.

#### Returns:

Image Data Related Error Codes, ROI Related Error Codes.

### 7.5.2.41 `NppStatus nppiThreshold_GT_32f_C3IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])`

3 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.42** `NppStatus nppiThreshold_GT_32f_C3R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])`

3 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is greater than 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.43** `NppStatus nppiThreshold_GT_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])`

4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

#### 7.5.2.44 `NppStatus nppiThreshold_GT_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])`

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

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set 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).
- rThresholds* The threshold values, one per color channel.

##### Returns:

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

#### 7.5.2.45 `NppStatus nppiThreshold_GT_8u_C1IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold)`

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

##### Parameters:

- pSrcDst* In-Place Image Pointer.
- nSrcDstStep* In-Place-Image Line Step.
- oSizeROI* Region-of-Interest (ROI).
- nThreshold* The threshold value.

##### Returns:

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

#### 7.5.2.46 `NppStatus nppiThreshold_GT_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold)`

1 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is greater than 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.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.47 NppStatus nppiThreshold\_GT\_8u\_C3IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])**

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.48 NppStatus nppiThreshold\_GT\_8u\_C3R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])**

3 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is greater than 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).

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.49** `NppStatus nppiThreshold_GTVal_16s_AC4IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])`

4 channel 16-bit signed short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.50** `NppStatus nppiThreshold_GTVal_16s_AC4R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])`

4 channel 16-bit signed short image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.51** `NppStatus nppiThreshold_GTVal_16s_C11R (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue)`

1 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.  
*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.52** `NppStatus nppiThreshold_GTVVal_16s_C1R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue)`

1 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, 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.  
*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.53** `NppStatus nppiThreshold_GTVVal_16s_C3IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])`

3 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.54 NppStatus nppiThreshold\_GTVAl\_16s\_C3R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])**

3 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.55 NppStatus nppiThreshold\_GTVAl\_16u\_AC4IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])**

4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.56 NppStatus nppiThreshold\_GTVAl\_16u\_AC4R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])**

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

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, 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).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.57 `NppStatus nppiThreshold_GTVVal_16u_C1IR(Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue)`

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.  
*nValue* The threshold replacement value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.58 `NppStatus nppiThreshold_GTVVal_16u_C1R(const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue)`

1 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.59** `NppStatus nppiThreshold_GTVAl_16u_C3IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.60** `NppStatus nppiThreshold_GTVAl_16u_C3R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

3 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.61 NppStatus nppiThreshold\_GTVal\_32f\_AC4IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])**

4 channel 32-bit floating point in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.62 NppStatus nppiThreshold\_GTVal\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])**

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

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.63 NppStatus nppiThreshold\_GTVal\_32f\_C11R (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold, const Npp32f nValue)**

1 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.  
*nValue* The threshold replacement values.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.64 **NppStatus nppiThreshold\_GTVal\_32f\_C1R** (const Npp32f \* *pSrc*, int *nSrcStep*, Npp32f \* *pDst*, int *nDstStep*, NppiSize *oSizeROI*, const Npp32f *nThreshold*, const Npp32f *nValue*)

1 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, 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.  
*nValue* The threshold replacement value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.65 **NppStatus nppiThreshold\_GTVal\_32f\_C3IR** (Npp32f \* *pSrcDst*, int *nSrcDstStep*, NppiSize *oSizeROI*, const Npp32f *rThresholds*[3], const Npp32f *rValues*[3])

3 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.66** `NppStatus nppiThreshold_GTVAl_32f_C3R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])`

3 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.67** `NppStatus nppiThreshold_GTVAl_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.68** `NppStatus nppiThreshold_GTVAl_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

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

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set value is set to rValue, 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).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.69 **NppStatus nppiThreshold\_GTVAl\_8u\_C1IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue)**

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.  
*nValue* The threshold replacement value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.70 **NppStatus nppiThreshold\_GTVAl\_8u\_C1R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue)**

1 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is greater than nThreshold is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.71 NppStatus nppiThreshold\_GTVVal\_8u\_C3IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])**

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.72 NppStatus nppiThreshold\_GTVVal\_8u\_C3R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])**

3 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is greater than rThreshold is true, the pixel is set to rValue, 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

### 7.5.2.73 `NppStatus nppiThreshold_LT_16s_AC4IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])`

4 channel 16-bit signed short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

- pSrcDst* In-Place Image Pointer.
- nSrcDstStep* In-Place-Image Line Step.
- oSizeROI* Region-of-Interest (ROI).
- rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.74 `NppStatus nppiThreshold_LT_16s_AC4R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])`

4 channel 16-bit signed short image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set 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).
- rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.75 `NppStatus nppiThreshold_LT_16s_C1IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold)`

1 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

- pSrcDst* In-Place Image Pointer.
- nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.76** `NppStatus nppiThreshold_LT_16s_C1R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold)`

1 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than 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.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.77** `NppStatus nppiThreshold_LT_16s_C3IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])`

3 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.



**7.5.2.78 NppStatus nppiThreshold\_LT\_16s\_C3R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3])**

3 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than 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).

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.79 NppStatus nppiThreshold\_LT\_16u\_AC4IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])**

4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.80 NppStatus nppiThreshold\_LT\_16u\_AC4R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])**

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

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.81 **NppStatus nppiThreshold\_LT\_16u\_C1IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold)**

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nThreshold* The threshold value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.82 **NppStatus nppiThreshold\_LT\_16u\_C1R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold)**

1 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than 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.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

### 7.5.2.83 `NppStatus nppiThreshold_LT_16u_C3IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])`

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.84 `NppStatus nppiThreshold_LT_16u_C3R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3])`

3 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than 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).

*rThresholds* The threshold values, one per color channel.

#### Returns:

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

### 7.5.2.85 `NppStatus nppiThreshold_LT_32f_AC4IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])`

4 channel 32-bit floating point in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

#### Parameters:

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.86 NppStatus nppiThreshold\_LT\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])**

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

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set 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).

*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.87 NppStatus nppiThreshold\_LT\_32f\_C1IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold)**

1 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.88 NppStatus nppiThreshold\_LT\_32f\_C1R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f nThreshold)**

1 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than 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.

**Returns:**

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

**7.5.2.89 NppStatus nppiThreshold\_LT\_32f\_C3IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])**

3 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

- pSrcDst* In-Place Image Pointer.
- nSrcDstStep* In-Place-Image Line Step.
- oSizeROI* Region-of-Interest (ROI).
- rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.90 NppStatus nppiThreshold\_LT\_32f\_C3R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3])**

3 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.91 `NppStatus nppiThreshold_LT_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])`

4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set value is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.92 `NppStatus nppiThreshold_LT_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])`

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

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set 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).  
*rThresholds* The threshold values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

**7.5.2.93 NppStatus nppiThreshold\_LT\_8u\_C1IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold)**

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

**Returns:**

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

**7.5.2.94 NppStatus nppiThreshold\_LT\_8u\_C1R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold)**

1 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than 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.

**Returns:**

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

**7.5.2.95 NppStatus nppiThreshold\_LT\_8u\_C3IR (Npp8u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])**

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nThreshold, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.96 NppStatus nppiThreshold\_LT\_8u\_C3R (const Npp8u \* pSrc, int nSrcStep, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3])**

3 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than 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).

*rThresholds* The threshold values, one per color channel.

**Returns:**

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

**7.5.2.97 NppStatus nppiThreshold\_LTVal\_16s\_AC4IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])**

4 channel 16-bit signed short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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



**7.5.2.98 NppStatus nppiThreshold\_LTVal\_16s\_AC4R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])**

4 channel 16-bit signed short image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.99 NppStatus nppiThreshold\_LTVal\_16s\_C1IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue)**

1 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.100 NppStatus nppiThreshold\_LTVal\_16s\_C1R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue)**

1 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, 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.  
*nValue* The threshold replacement value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.101 NppStatus nppiThreshold\_LTVal\_16s\_C3IR (Npp16s \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])

3 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.102 NppStatus nppiThreshold\_LTVal\_16s\_C3R (const Npp16s \* pSrc, int nSrcStep, Npp16s \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3])

3 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.103** `NppStatus nppiThreshold_LTVal_16u_AC4IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.104** `NppStatus nppiThreshold_LTVal_16u_AC4R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

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

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.105** `NppStatus nppiThreshold_LTVal_16u_C1IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue)`

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nThreshold* The threshold value.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.106** `NppStatus nppiThreshold_LTVal_16u_C1R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue)`

1 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.107** `NppStatus nppiThreshold_LTVal_16u_C3IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.108** `NppStatus nppiThreshold_LTVal_16u_C3R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3])`

3 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, 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).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.109** `NppStatus nppiThreshold_LTVal_32f_AC4IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])`

4 channel 32-bit floating point in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.110** `NppStatus nppiThreshold_LTVAl_32f_AC4R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])`

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

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.111** `NppStatus nppiThreshold_LTVAl_32f_C1IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold, const Npp32f nValue)`

1 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.112** `NppStatus nppiThreshold_LTVAl_32f_C1R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f nThreshold, const Npp32f nValue)`

1 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, 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.  
*nValue* The threshold replacement value.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.113 `NppStatus nppiThreshold_LTVal_32f_C3IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])`

3 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.  
*nSrcDstStep* In-Place-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*rThresholds* The threshold values, one per color channel.  
*rValues* The threshold replacement values, one per color channel.

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes.

#### 7.5.2.114 `NppStatus nppiThreshold_LTVal_32f_C3R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3])`

3 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.115** `NppStatus nppiThreshold_LTVal_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.116** `NppStatus nppiThreshold_LTVal_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

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

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set value is set to rValue, 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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



**7.5.2.117** `NppStatus nppiThreshold_LTVal_8u_C1IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue)`

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.118** `NppStatus nppiThreshold_LTVal_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue)`

1 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than nThreshold is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement value.

**Returns:**

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

**7.5.2.119** `NppStatus nppiThreshold_LTVal_8u_C3IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.120** `NppStatus nppiThreshold_LTVal_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3])`

3 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than rThreshold is true, the pixel is set to rValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement values, one per color channel.

**Returns:**

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

**7.5.2.121** `NppStatus nppiThreshold_LTValGTVal_16s_AC4IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholdsLT[3], const Npp16s rValuesLT[3], const Npp16s rThresholdsGT[3], const Npp16s rValuesGT[3])`

4 channel 16-bit signed short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.122** `NppStatus nppiThreshold_LTValGTVal_16s_AC4R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholdsLT[3], const Npp16s rValuesLT[3], const Npp16s rThresholdsGT[3], const Npp16s rValuesGT[3])`

4 channel 16-bit signed short image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.123** `NppStatus nppiThreshold_LTValGTVal_16s_C1IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThresholdLT, const Npp16s nValueLT, const Npp16s nThresholdGT, const Npp16s nValueGT)`

1 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.

*nThresholdGT* The thresholdGT value.

*nValueGT* The thresholdGT replacement value.

**Returns:**

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

**7.5.2.124** `NppStatus nppiThreshold_LTValGTVal_16s_C1R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThresholdLT, const Npp16s nValueLT, const Npp16s nThresholdGT, const Npp16s nValueGT)`

1 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, 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).

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.

*nThresholdGT* The thresholdGT value.

*nValueGT* The thresholdGT replacement value.

**Returns:**

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

**7.5.2.125** `NppStatus nppiThreshold_LTValGTVal_16s_C3IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholdsLT[3], const Npp16s rValuesLT[3], const Npp16s rThresholdsGT[3], const Npp16s rValuesGT[3])`

3 channel 16-bit signed short in place threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.126** `NppStatus nppiThreshold_LTValGTVal_16s_C3R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholdsLT[3], const Npp16s rValuesLT[3], const Npp16s rThresholdsGT[3], const Npp16s rValuesGT[3])`

3 channel 16-bit signed short threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.127** `NppStatus nppiThreshold_LTValGTVal_16u_AC4IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholdsLT[3], const Npp16u rValuesLT[3], const Npp16u rThresholdsGT[3], const Npp16u rValuesGT[3])`

4 channel 16-bit unsigned short in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.128** `NppStatus nppiThreshold_LTValGTVal_16u_AC4R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholdsLT[3], const Npp16u rValuesLT[3], const Npp16u rThresholdsGT[3], const Npp16u rValuesGT[3])`

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

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.129** `NppStatus nppiThreshold_LTValGTVal_16u_C1IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThresholdLT, const Npp16u nValueLT, const Npp16u nThresholdGT, const Npp16u nValueGT)`

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.

*nThresholdGT* The thresholdGT value.

*nValueGT* The thresholdGT replacement value.

**Returns:**

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

**7.5.2.130 NppStatus nppiThreshold\_LTValGTVal\_16u\_C1R (const Npp16u \* pSrc, int nSrcStep, Npp16u \* pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThresholdLT, const Npp16u nValueLT, const Npp16u nThresholdGT, const Npp16u nValueGT)**

1 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, 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).

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.

*nThresholdGT* The thresholdGT value.

*nValueGT* The thresholdGT replacement value.

**Returns:**

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

**7.5.2.131 NppStatus nppiThreshold\_LTValGTVal\_16u\_C3IR (Npp16u \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholdsLT[3], const Npp16u rValuesLT[3], const Npp16u rThresholdsGT[3], const Npp16u rValuesGT[3])**

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.132** `NppStatus nppiThreshold_LTValGTVal_16u_C3R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholdsLT[3], const Npp16u rValuesLT[3], const Npp16u rThresholdsGT[3], const Npp16u rValuesGT[3])`

3 channel 16-bit unsigned short threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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 \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.133** `NppStatus nppiThreshold_LTValGTVal_32f_AC4IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholdsLT[3], const Npp32f rValuesLT[3], const Npp32f rThresholdsGT[3], const Npp32f rValuesGT[3])`

4 channel 32-bit floating point in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.



*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.134 NppStatus nppiThreshold\_LTValGTVal\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholdsLT[3], const Npp32f rValuesLT[3], const Npp32f rThresholdsGT[3], const Npp32f rValuesGT[3])**

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

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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 \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.135 NppStatus nppiThreshold\_LTValGTVal\_32f\_C1IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThresholdLT, const Npp32f nValueLT, const Npp32f nThresholdGT, const Npp32f nValueGT)**

1 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.  
*nThresholdGT* The thresholdGT value.  
*nValueGT* The thresholdGT replacement value.

**Returns:**

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

### 7.5.2.136 NppStatus nppiThreshold\_LTValGTVal\_32f\_C1R (const Npp32f \* pSrc, int nSrcStep, Npp32f \* pDst, int nDstStep, NppiSize oSizeROI, const Npp32f nThresholdLT, const Npp32f nValueLT, const Npp32f nThresholdGT, const Npp32f nValueGT)

1 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, 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 \(ROI\)](#).  
*nThresholdLT* The thresholdLT value.  
*nValueLT* The thresholdLT replacement value.  
*nThresholdGT* The thresholdGT value.  
*nValueGT* The thresholdGT replacement value.

**Returns:**

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

### 7.5.2.137 NppStatus nppiThreshold\_LTValGTVal\_32f\_C3IR (Npp32f \* pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholdsLT[3], const Npp32f rValuesLT[3], const Npp32f rThresholdsGT[3], const Npp32f rValuesGT[3])

3 channel 32-bit floating point in place threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).  
*nSrcDstStep* [In-Place-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.138** `NppStatus nppiThreshold_LTValGTVal_32f_C3R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholdsLT[3], const Npp32f rValuesLT[3], const Npp32f rThresholdsGT[3], const Npp32f rValuesGT[3])`

3 channel 32-bit floating point threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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 \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.139** `NppStatus nppiThreshold_LTValGTVal_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholdsLT[3], const Npp8u rValuesLT[3], const Npp8u rThresholdsGT[3], const Npp8u rValuesGT[3])`

4 channel 8-bit unsigned char in place image threshold, not affecting Alpha.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.140** `NppStatus nppiThreshold_LTValGTVal_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholdsLT[3], const Npp8u rValuesLT[3], const Npp8u rThresholdsGT[3], const Npp8u rValuesGT[3])`

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

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set value is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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 \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.141** `NppStatus nppiThreshold_LTValGTVal_8u_C1IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThresholdLT, const Npp8u nValueLT, const Npp8u nThresholdGT, const Npp8u nValueGT)`

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*nThresholdLT* The thresholdLT value.

*nValueLT* The thresholdLT replacement value.  
*nThresholdGT* The thresholdGT value.  
*nValueGT* The thresholdGT replacement value.

**Returns:**

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

#### 7.5.2.142 `NppStatus nppiThreshold_LTValGTVal_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThresholdLT, const Npp8u nValueLT, const Npp8u nThresholdGT, const Npp8u nValueGT)`

1 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than nThresholdLT is true, the pixel is set to nValueLT, else if sourcePixel is greater than nThresholdGT the pixel is set to nValueGT, 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 \(ROI\)](#).  
*nThresholdLT* The thresholdLT value.  
*nValueLT* The thresholdLT replacement value.  
*nThresholdGT* The thresholdGT value.  
*nValueGT* The thresholdGT replacement value.

**Returns:**

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

#### 7.5.2.143 `NppStatus nppiThreshold_LTValGTVal_8u_C3IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholdsLT[3], const Npp8u rValuesLT[3], const Npp8u rThresholdsGT[3], const Npp8u rValuesGT[3])`

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [Destination-Image Pointer](#).  
*nSrcDstStep* [Destination-Image Line Step](#).  
*oSizeROI* [Region-of-Interest \(ROI\)](#).  
*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.144** `NppStatus nppiThreshold_LTValGTVal_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholdsLT[3], const Npp8u rValuesLT[3], const Npp8u rThresholdsGT[3], const Npp8u rValuesGT[3])`

3 channel 8-bit unsigned char threshold.

If for a comparison operations sourcePixel is less than rThresholdLT is true, the pixel is set to rValueLT, else if sourcePixel is greater than rThresholdGT the pixel is set to rValueGT, 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 \(ROI\)](#).

*rThresholdsLT* The thresholdLT values, one per color channel.

*rValuesLT* The thresholdLT replacement values, one per color channel.

*rThresholdsGT* The thresholdGT values, one per channel.

*rValuesGT* The thresholdGT replacement values, one per color channel.

**Returns:**

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

**7.5.2.145** `NppStatus nppiThreshold_Val_16s_AC4IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit signed short in place 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 nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.146** `NppStatus nppiThreshold_Val_16s_AC4R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit signed short 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 nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.147** `NppStatus nppiThreshold_Val_16s_C1IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue, NppCmpOp eComparisonOperation)`

1 channel 16-bit signed short in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement 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.

**7.5.2.148** `NppStatus nppiThreshold_Val_16s_C1R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s nThreshold, const Npp16s nValue, NppCmpOp eComparisonOperation)`

1 channel 16-bit signed short threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement 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.

**7.5.2.149** `NppStatus nppiThreshold_Val_16s_C3IR (Npp16s * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3], NppCmpOp eComparisonOperation)`

3 channel 16-bit signed short in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.150** `NppStatus nppiThreshold_Val_16s_C3R (const Npp16s * pSrc, int nSrcStep, Npp16s * pDst, int nDstStep, NppiSize oSizeROI, const Npp16s rThresholds[3], const Npp16s rValues[3], NppCmpOp eComparisonOperation)`

3 channel 16-bit signed short threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.151** `NppStatus nppiThreshold_Val_16u_AC4IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit unsigned short in place 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 nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.152** `NppStatus nppiThreshold_Val_16u_AC4R (const Npp16u *pSrc, int nSrcStep, Npp16u *pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3], NppCmpOp eComparisonOperation)`

4 channel 16-bit unsigned short 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 nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.153** `NppStatus nppiThreshold_Val_16u_C1IR (Npp16u *pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue, NppCmpOp eComparisonOperation)`

1 channel 16-bit unsigned short in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement 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.

**7.5.2.154** `NppStatus nppiThreshold_Val_16u_C1R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u nThreshold, const Npp16u nValue, NppCmpOp eComparisonOperation)`

1 channel 16-bit unsigned short threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement 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.

**7.5.2.155** `NppStatus nppiThreshold_Val_16u_C3IR (Npp16u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3], NppCmpOp eComparisonOperation)`

3 channel 16-bit unsigned short in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.156** `NppStatus nppiThreshold_Val_16u_C3R (const Npp16u * pSrc, int nSrcStep, Npp16u * pDst, int nDstStep, NppiSize oSizeROI, const Npp16u rThresholds[3], const Npp16u rValues[3], NppCmpOp eComparisonOperation)`

3 channel 16-bit unsigned short threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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 \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.157** `NppStatus nppiThreshold_Val_32f_AC4IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3], NppCmpOp eComparisonOperation)`

4 channel 32-bit floating point in place 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 nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* [In-Place Image Pointer](#).

*nSrcDstStep* [In-Place-Image Line Step](#).

*oSizeROI* [Region-of-Interest \(ROI\)](#).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.158** `NppStatus nppiThreshold_Val_32f_AC4R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3], NppCmpOp eComparisonOperation)`

4 channel 32-bit floating point 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 nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.159** `NppStatus nppiThreshold_Val_32f_C1IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f nThreshold, const Npp32f nValue, NppCmpOp eComparisonOperation)`

1 channel 32-bit floating point in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement 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.

**7.5.2.160** `NppStatus nppiThreshold_Val_32f_C1R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f nThreshold, const Npp32f nValue, NppCmpOp eComparisonOperation)`

1 channel 32-bit floating point threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement 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.

**7.5.2.161** `NppStatus nppiThreshold_Val_32f_C3IR (Npp32f * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3], NppCmpOp eComparisonOperation)`

3 channel 32-bit floating point in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.162** `NppStatus nppiThreshold_Val_32f_C3R (const Npp32f * pSrc, int nSrcStep, Npp32f * pDst, int nDstStep, NppiSize oSizeROI, const Npp32f rThresholds[3], const Npp32f rValues[3], NppCmpOp eComparisonOperation)`

3 channel 32-bit floating point threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.163** `NppStatus nppiThreshold_Val_8u_AC4IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3], NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned char in place 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 nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.164** `NppStatus nppiThreshold_Val_8u_AC4R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3], NppCmpOp eComparisonOperation)`

4 channel 8-bit unsigned char 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 nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.165** `NppStatus nppiThreshold_Val_8u_C1IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue, NppCmpOp eComparisonOperation)`

1 channel 8-bit unsigned char in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*nThreshold* The threshold value.

*nValue* The threshold replacement 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.

**7.5.2.166** `NppStatus nppiThreshold_Val_8u_C1R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u nThreshold, const Npp8u nValue, NppCmpOp eComparisonOperation)`

1 channel 8-bit unsigned char threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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.

*nValue* The threshold replacement 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.

**7.5.2.167** `NppStatus nppiThreshold_Val_8u_C3IR (Npp8u * pSrcDst, int nSrcDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3], NppCmpOp eComparisonOperation)`

3 channel 8-bit unsigned char in place threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, otherwise it is set to sourcePixel.

**Parameters:**

*pSrcDst* In-Place Image Pointer.

*nSrcDstStep* In-Place-Image Line Step.

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

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

**7.5.2.168** `NppStatus nppiThreshold_Val_8u_C3R (const Npp8u * pSrc, int nSrcStep, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, const Npp8u rThresholds[3], const Npp8u rValues[3], NppCmpOp eComparisonOperation)`

3 channel 8-bit unsigned char threshold.

If for a comparison operations OP the predicate (sourcePixel OP nThreshold) is true, the pixel is set to nValue, 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).

*rThresholds* The threshold values, one per color channel.

*rValues* The threshold replacement 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.

## 7.6 Compare Operations

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

### Functions

- `NppStatus nppiCompare_8u_C1R` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*1 channel 8-bit unsigned char image compare.*
- `NppStatus nppiCompare_8u_C3R` (const `Npp8u` \*pSrc1, int nSrc1Step, const `Npp8u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*3 channel 8-bit unsigned char image compare.*
- `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 char 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 char image compare, not affecting Alpha.*
- `NppStatus nppiCompare_16u_C1R` (const `Npp16u` \*pSrc1, int nSrc1Step, const `Npp16u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit unsigned short image compare.*
- `NppStatus nppiCompare_16u_C3R` (const `Npp16u` \*pSrc1, int nSrc1Step, const `Npp16u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit unsigned short image compare.*
- `NppStatus nppiCompare_16u_C4R` (const `Npp16u` \*pSrc1, int nSrc1Step, const `Npp16u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*4 channel 16-bit unsigned short image compare.*
- `NppStatus nppiCompare_16u_AC4R` (const `Npp16u` \*pSrc1, int nSrc1Step, const `Npp16u` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*4 channel 16-bit unsigned short image compare, not affecting Alpha.*
- `NppStatus nppiCompare_16s_C1R` (const `Npp16s` \*pSrc1, int nSrc1Step, const `Npp16s` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*1 channel 16-bit signed short image compare.*
- `NppStatus nppiCompare_16s_C3R` (const `Npp16s` \*pSrc1, int nSrc1Step, const `Npp16s` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*3 channel 16-bit signed short image compare.*
- `NppStatus nppiCompare_16s_C4R` (const `Npp16s` \*pSrc1, int nSrc1Step, const `Npp16s` \*pSrc2, int nSrc2Step, `Npp8u` \*pDst, int nDstStep, `NppiSize` oSizeROI, `NppCmpOp` eComparisonOperation)  
*4 channel 16-bit signed short image compare.*

- [NppStatus nppiCompare\\_16s\\_AC4R](#) (const [Npp16s](#) \*pSrc1, int nSrc1Step, const [Npp16s](#) \*pSrc2, int nSrc2Step, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 16-bit signed short 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)  
*1 channel 32-bit floating point image compare.*
- [NppStatus nppiCompare\\_32f\\_C3R](#) (const [Npp32f](#) \*pSrc1, int nSrc1Step, const [Npp32f](#) \*pSrc2, int nSrc2Step, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*3 channel 32-bit floating point image compare.*
- [NppStatus nppiCompare\\_32f\\_C4R](#) (const [Npp32f](#) \*pSrc1, int nSrc1Step, const [Npp32f](#) \*pSrc2, int nSrc2Step, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 32-bit floating point image compare.*
- [NppStatus nppiCompare\\_32f\\_AC4R](#) (const [Npp32f](#) \*pSrc1, int nSrc1Step, const [Npp32f](#) \*pSrc2, int nSrc2Step, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 32-bit signed floating point compare, not affecting Alpha.*
- [NppStatus nppiCompareC\\_8u\\_C1R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, const [Npp8u](#) nConstant, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*1 channel 8-bit unsigned char image compare with constant value.*
- [NppStatus nppiCompareC\\_8u\\_C3R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, const [Npp8u](#) \*pConstants, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*3 channel 8-bit unsigned char image compare with constant value.*
- [NppStatus nppiCompareC\\_8u\\_C4R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, const [Npp8u](#) \*pConstants, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 8-bit unsigned char image compare with constant value.*
- [NppStatus nppiCompareC\\_8u\\_AC4R](#) (const [Npp8u](#) \*pSrc, int nSrcStep, const [Npp8u](#) \*pConstants, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 8-bit unsigned char image compare, not affecting Alpha.*
- [NppStatus nppiCompareC\\_16u\\_C1R](#) (const [Npp16u](#) \*pSrc, int nSrcStep, const [Npp16u](#) nConstant, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*1 channel 16-bit unsigned short image compare with constant value.*
- [NppStatus nppiCompareC\\_16u\\_C3R](#) (const [Npp16u](#) \*pSrc, int nSrcStep, const [Npp16u](#) \*pConstants, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*3 channel 16-bit unsigned short image compare with constant value.*
- [NppStatus nppiCompareC\\_16u\\_C4R](#) (const [Npp16u](#) \*pSrc, int nSrcStep, const [Npp16u](#) \*pConstants, [Npp8u](#) \*pDst, int nDstStep, [NppiSize](#) oSizeROI, [NppCmpOp](#) eComparisonOperation)  
*4 channel 16-bit unsigned short image compare with constant value.*

- **NppStatus nppiCompareC\_16u\_AC4R** (const **Npp16u** \*pSrc, int nSrcStep, const **Npp16u** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*4 channel 16-bit unsigned short image compare, not affecting Alpha.*
- **NppStatus nppiCompareC\_16s\_C1R** (const **Npp16s** \*pSrc, int nSrcStep, const **Npp16s** nConstant, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*1 channel 16-bit signed short image compare with constant value.*
- **NppStatus nppiCompareC\_16s\_C3R** (const **Npp16s** \*pSrc, int nSrcStep, const **Npp16s** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*3 channel 16-bit signed short image compare with constant value.*
- **NppStatus nppiCompareC\_16s\_C4R** (const **Npp16s** \*pSrc, int nSrcStep, const **Npp16s** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*4 channel 16-bit signed short image compare with constant value.*
- **NppStatus nppiCompareC\_16s\_AC4R** (const **Npp16s** \*pSrc, int nSrcStep, const **Npp16s** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*4 channel 16-bit signed short image compare, not affecting Alpha.*
- **NppStatus nppiCompareC\_32f\_C1R** (const **Npp32f** \*pSrc, int nSrcStep, const **Npp32f** nConstant, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*1 channel 32-bit floating point image compare with constant value.*
- **NppStatus nppiCompareC\_32f\_C3R** (const **Npp32f** \*pSrc, int nSrcStep, const **Npp32f** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*3 channel 32-bit floating point image compare with constant value.*
- **NppStatus nppiCompareC\_32f\_C4R** (const **Npp32f** \*pSrc, int nSrcStep, const **Npp32f** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*4 channel 32-bit floating point image compare with constant value.*
- **NppStatus nppiCompareC\_32f\_AC4R** (const **Npp32f** \*pSrc, int nSrcStep, const **Npp32f** \*pConstants, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **NppCmpOp** eComparisonOperation)  
*4 channel 32-bit signed floating point compare, not affecting Alpha.*
- **NppStatus nppiCompareEqualEps\_32f\_C1R** (const **Npp32f** \*pSrc1, int nSrc1Step, const **Npp32f** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **Npp32f** nEpsilon)  
*1 channel 32-bit floating point image compare whether two images are equal within epsilon.*
- **NppStatus nppiCompareEqualEps\_32f\_C3R** (const **Npp32f** \*pSrc1, int nSrc1Step, const **Npp32f** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **Npp32f** nEpsilon)  
*3 channel 32-bit floating point image compare whether two images are equal within epsilon.*
- **NppStatus nppiCompareEqualEps\_32f\_C4R** (const **Npp32f** \*pSrc1, int nSrc1Step, const **Npp32f** \*pSrc2, int nSrc2Step, **Npp8u** \*pDst, int nDstStep, **NppiSize** oSizeROI, **Npp32f** nEpsilon)  
*4 channel 32-bit floating point image compare whether two images are equal within epsilon.*

- `NppStatus nppiCompareEqualEps_32f_AC4R` (const `Npp32f *pSrc1`, int `nSrc1Step`, const `Npp32f *pSrc2`, int `nSrc2Step`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `Npp32f nEpsilon`)  
*4 channel 32-bit signed floating point compare whether two images are equal within epsilon, not affecting Alpha.*
- `NppStatus nppiCompareEqualEpsC_32f_C1R` (const `Npp32f *pSrc`, int `nSrcStep`, const `Npp32f nConstant`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `Npp32f nEpsilon`)  
*1 channel 32-bit floating point image compare whether image and constant are equal within epsilon.*
- `NppStatus nppiCompareEqualEpsC_32f_C3R` (const `Npp32f *pSrc`, int `nSrcStep`, const `Npp32f *pConstants`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `Npp32f nEpsilon`)  
*3 channel 32-bit floating point image compare whether image and constant are equal within epsilon.*
- `NppStatus nppiCompareEqualEpsC_32f_C4R` (const `Npp32f *pSrc`, int `nSrcStep`, const `Npp32f *pConstants`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `Npp32f nEpsilon`)  
*4 channel 32-bit floating point image compare whether image and constant are equal within epsilon.*
- `NppStatus nppiCompareEqualEpsC_32f_AC4R` (const `Npp32f *pSrc`, int `nSrcStep`, const `Npp32f *pConstants`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `Npp32f nEpsilon`)  
*4 channel 32-bit signed floating point compare whether image and constant are equal within epsilon, not affecting Alpha.*

## 7.6.1 Detailed Description

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`.

## 7.6.2 Function Documentation

**7.6.2.1** `NppStatus nppiCompare_16s_AC4R` (const `Npp16s *pSrc1`, int `nSrc1Step`, const `Npp16s *pSrc2`, int `nSrc2Step`, `Npp8u *pDst`, int `nDstStep`, `NppiSize oSizeROI`, `NppCmpOp eComparisonOperation`)

4 channel 16-bit signed short 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](#)

**7.6.2.2** `NppStatus nppiCompare_16s_C1R (const Npp16s * pSrc1, int nSrc1Step, const Npp16s * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)`

1 channel 16-bit signed short 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](#)

**7.6.2.3** `NppStatus nppiCompare_16s_C3R (const Npp16s * pSrc1, int nSrc1Step, const Npp16s * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)`

3 channel 16-bit signed short 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](#)

#### 7.6.2.4 NppStatus nppiCompare\_16s\_C4R (const Npp16s \* pSrc1, int nSrc1Step, const Npp16s \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)

4 channel 16-bit signed short 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](#)

#### 7.6.2.5 NppStatus nppiCompare\_16u\_AC4R (const Npp16u \* pSrc1, int nSrc1Step, const Npp16u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)

4 channel 16-bit unsigned short 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](#)



**7.6.2.6 NppStatus nppiCompare\_16u\_C1R (const Npp16u \* pSrc1, int nSrc1Step, const Npp16u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 16-bit unsigned short 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](#)

**7.6.2.7 NppStatus nppiCompare\_16u\_C3R (const Npp16u \* pSrc1, int nSrc1Step, const Npp16u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 16-bit unsigned short 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](#)

**7.6.2.8 NppStatus nppiCompare\_16u\_C4R (const Npp16u \* pSrc1, int nSrc1Step, const Npp16u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 16-bit unsigned short 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](#)

**7.6.2.9 NppStatus nppiCompare\_32f\_AC4R (const Npp32f \* pSrc1, int nSrc1Step, const Npp32f \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 32-bit signed floating point 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](#)

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

1 channel 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](#)

**7.6.2.11 NppStatus nppiCompare\_32f\_C3R (const Npp32f \* pSrc1, int nSrc1Step, const Npp32f \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 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](#)

**7.6.2.12 NppStatus nppiCompare\_32f\_C4R (const Npp32f \* pSrc1, int nSrc1Step, const Npp32f \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 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](#)

**7.6.2.13 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 char 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](#)

**7.6.2.14 NppStatus nppiCompare\_8u\_C1R (const Npp8u \* pSrc1, int nSrc1Step, const Npp8u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 8-bit unsigned char 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](#)

**7.6.2.15 NppStatus nppiCompare\_8u\_C3R (const Npp8u \* pSrc1, int nSrc1Step, const Npp8u \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 8-bit unsigned char 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](#)

**7.6.2.16 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 char 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](#)

**7.6.2.17 NppStatus nppiCompareC\_16s\_AC4R (const Npp16s \* pSrc, int nSrcStep, const Npp16s \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 16-bit signed short image compare, not affecting Alpha.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.18 NppStatus nppiCompareC\_16s\_C1R (const Npp16s \* pSrc, int nSrcStep, const Npp16s nConstant, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 16-bit signed short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- nConstant* constant value.
- 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

#### 7.6.2.19 NppStatus nppiCompareC\_16s\_C3R (const Npp16s \* pSrc, int nSrcStep, const Npp16s \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)

3 channel 16-bit signed short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- pConstants* pointer to a list of constants, one per color channel.
- 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

#### 7.6.2.20 NppStatus nppiCompareC\_16s\_C4R (const Npp16s \* pSrc, int nSrcStep, const Npp16s \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)

4 channel 16-bit signed short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- pConstants* pointer to a list of constants, one per color channel.
- 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](#)

**7.6.2.21 NppStatus nppiCompareC\_16u\_AC4R (const Npp16u \* pSrc, int nSrcStep, const Npp16u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

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

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.22 NppStatus nppiCompareC\_16u\_C1R (const Npp16u \* pSrc, int nSrcStep, const Npp16u nConstant, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 16-bit unsigned short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*nConstant* constant value

*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](#)



**7.6.2.23 NppStatus nppiCompareC\_16u\_C3R (const Npp16u \* pSrc, int nSrcStep, const Npp16u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 16-bit unsigned short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.24 NppStatus nppiCompareC\_16u\_C4R (const Npp16u \* pSrc, int nSrcStep, const Npp16u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 16-bit unsigned short image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.25 NppStatus nppiCompareC\_32f\_AC4R (const Npp32f \* pSrc, int nSrcStep, const Npp32f \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 32-bit signed floating point compare, not affecting Alpha.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- pConstants* pointer to a list of constants, one per color channel.
- 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

#### 7.6.2.26 **NppStatus nppiCompareC\_32f\_C1R (const Npp32f \* pSrc, int nSrcStep, const Npp32f nConstant, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 32-bit floating point image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- nConstant* constant value
- 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

#### 7.6.2.27 **NppStatus nppiCompareC\_32f\_C3R (const Npp32f \* pSrc, int nSrcStep, const Npp32f \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 32-bit floating point image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- pConstants* pointer to a list of constants, one per color channel.
- 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](#)

**7.6.2.28 NppStatus nppiCompareC\_32f\_C4R (const Npp32f \* pSrc, int nSrcStep, const Npp32f \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 32-bit floating point image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.29 NppStatus nppiCompareC\_8u\_AC4R (const Npp8u \* pSrc, int nSrcStep, const Npp8u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

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

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*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](#)

**7.6.2.30 NppStatus nppiCompareC\_8u\_C1R (const Npp8u \* pSrc, int nSrcStep, const Npp8u nConstant, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

1 channel 8-bit unsigned char image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*nConstant* constant value.

*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](#)

**7.6.2.31 NppStatus nppiCompareC\_8u\_C3R (const Npp8u \* pSrc, int nSrcStep, const Npp8u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

3 channel 8-bit unsigned char image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constant values, one per color channel..

*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](#)

**7.6.2.32 NppStatus nppiCompareC\_8u\_C4R (const Npp8u \* pSrc, int nSrcStep, const Npp8u \* pConstants, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, NppCmpOp eComparisonOperation)**

4 channel 8-bit unsigned char image compare with constant value.

Compare pSrc's pixels with constant value.

**Parameters:**

- pSrc* Source-Image Pointer.
- nSrcStep* Source-Image Line Step.
- pConstants* pointer to a list of constants, one per color channel.
- 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

### 7.6.2.33 NppStatus nppiCompareEqualEps\_32f\_AC4R (const Npp32f \* pSrc1, int nSrc1Step, const Npp32f \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)

4 channel 32-bit signed floating point compare whether two images are equal within epsilon, not affecting Alpha.

Compare pSrc1's pixels with corresponding pixels in pSrc2 to determine whether they are equal with a difference of epsilon.

**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).
- nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

Image Data Related Error Codes, ROI Related Error Codes

### 7.6.2.34 NppStatus nppiCompareEqualEps\_32f\_C1R (const Npp32f \* pSrc1, int nSrc1Step, const Npp32f \* pSrc2, int nSrc2Step, Npp8u \* pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)

1 channel 32-bit floating point image compare whether two images are equal within epsilon.

Compare pSrc1's pixels with corresponding pixels in pSrc2 to determine whether they are equal with a difference of epsilon.

**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).  
*nEpsilon* epsilon tolerance value to compare to pixel absolute differences

**Returns:**

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

#### 7.6.2.35 `NppStatus nppiCompareEqualEps_32f_C3R (const Npp32f * pSrc1, int nSrc1Step, const Npp32f * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

3 channel 32-bit floating point image compare whether two images are equal within epsilon.

Compare pSrc1's pixels with corresponding pixels in pSrc2 to determine whether they are equal with a difference of epsilon.

**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).  
*nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

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

#### 7.6.2.36 `NppStatus nppiCompareEqualEps_32f_C4R (const Npp32f * pSrc1, int nSrc1Step, const Npp32f * pSrc2, int nSrc2Step, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

4 channel 32-bit floating point image compare whether two images are equal within epsilon.

Compare pSrc1's pixels with corresponding pixels in pSrc2 to determine whether they are equal with a difference of epsilon.

**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).

*nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

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

**7.6.2.37** `NppStatus nppiCompareEqualEpsC_32f_AC4R (const Npp32f * pSrc, int nSrcStep, const Npp32f * pConstants, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

4 channel 32-bit signed floating point compare whether image and constant are equal within epsilon, not affecting Alpha.

Compare pSrc's pixels with constant value to determine whether they are equal within a difference of epsilon.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*pConstants* pointer to a list of constants, one per color channel.

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.

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

*nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

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

**7.6.2.38** `NppStatus nppiCompareEqualEpsC_32f_C1R (const Npp32f * pSrc, int nSrcStep, const Npp32f nConstant, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

1 channel 32-bit floating point image compare whether image and constant are equal within epsilon.

Compare pSrc's pixels with constant value to determine whether they are equal within a difference of epsilon.

**Parameters:**

*pSrc* Source-Image Pointer.

*nSrcStep* Source-Image Line Step.

*nConstant* constant value

*pDst* Destination-Image Pointer.

*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nEpsilon* epsilon tolerance value to compare to pixel absolute differences

**Returns:**

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

### 7.6.2.39 `NppStatus nppiCompareEqualEpsC_32f_C3R (const Npp32f * pSrc, int nSrcStep, const Npp32f * pConstants, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

3 channel 32-bit floating point image compare whether image and constant are equal within epsilon.

Compare pSrc's pixels with constant value to determine whether they are equal within a difference of epsilon.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pConstants* pointer to a list of constants, one per color channel.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

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

### 7.6.2.40 `NppStatus nppiCompareEqualEpsC_32f_C4R (const Npp32f * pSrc, int nSrcStep, const Npp32f * pConstants, Npp8u * pDst, int nDstStep, NppiSize oSizeROI, Npp32f nEpsilon)`

4 channel 32-bit floating point image compare whether image and constant are equal within epsilon.

Compare pSrc's pixels with constant value to determine whether they are equal within a difference of epsilon.

**Parameters:**

*pSrc* Source-Image Pointer.  
*nSrcStep* Source-Image Line Step.  
*pConstants* pointer to a list of constants, one per color channel.  
*pDst* Destination-Image Pointer.  
*nDstStep* Destination-Image Line Step.  
*oSizeROI* Region-of-Interest (ROI).  
*nEpsilon* epsilon tolerance value to compare to per color channel pixel absolute differences

**Returns:**

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



# Chapter 8

## Data Structure Documentation

### 8.1 NPP\_ALIGN\_16 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.*
- [Npp64f re](#)  
*Real part.*
- [Npp64f im](#)  
*Imaginary part.*

#### 8.1.1 Detailed Description

Complex Number This struct represents a long long complex number.

Complex Number This struct represents a double floating-point complex number.

#### 8.1.2 Field Documentation

##### 8.1.2.1 Npp64f NPP\_ALIGN\_16::im

Imaginary part.

**8.1.2.2 Npp64s NPP\_ALIGN\_16::im**

Imaginary part.

**8.1.2.3 Npp64f NPP\_ALIGN\_16::re**

Real part.

**8.1.2.4 Npp64s NPP\_ALIGN\_16::re**

Real part.

The documentation for this struct was generated from the following file:

- C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h

## 8.2 NPP\_ALIGN\_8 Struct Reference

Complex Number This struct represents an unsigned int complex number.

```
#include <nppdefs.h>
```

### Data Fields

- [Npp32u re](#)  
*Real part.*
- [Npp32u im](#)  
*Imaginary part.*
- [Npp32s re](#)  
*Real part.*
- [Npp32s im](#)  
*Imaginary part.*
- [Npp32f re](#)  
*Real part.*
- [Npp32f im](#)  
*Imaginary part.*

### 8.2.1 Detailed Description

Complex Number This struct represents an unsigned int complex number.

Complex Number This struct represents a single floating-point complex number.

Complex Number This struct represents a signed int complex number.

### 8.2.2 Field Documentation

#### 8.2.2.1 Npp32f NPP\_ALIGN\_8::im

Imaginary part.

#### 8.2.2.2 Npp32s NPP\_ALIGN\_8::im

Imaginary part.

#### 8.2.2.3 Npp32u NPP\_ALIGN\_8::im

Imaginary part.

**8.2.2.4 Npp32f NPP\_ALIGN\_8::re**

Real part.

**8.2.2.5 Npp32s NPP\_ALIGN\_8::re**

Real part.

**8.2.2.6 Npp32u NPP\_ALIGN\_8::re**

Real part.

The documentation for this struct was generated from the following file:

- C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h

## 8.3 NppiHaarBuffer Struct Reference

```
#include <nppdefs.h>
```

### Data Fields

- int `haarBufferSize`  
*size of the buffer*
- `Npp32s * haarBuffer`  
*buffer*

### 8.3.1 Field Documentation

#### 8.3.1.1 `Npp32s* NppiHaarBuffer::haarBuffer`

`buffer`

#### 8.3.1.2 `int NppiHaarBuffer::haarBufferSize`

*size of the buffer*

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

## 8.4 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`

### 8.4.1 Field Documentation

#### 8.4.1.1 `Npp32s* NppiHaarClassifier_32f::classifiers`

packed classifier data 40 bytes each

#### 8.4.1.2 `NppiSize NppiHaarClassifier_32f::classifierSize`

#### 8.4.1.3 `size_t NppiHaarClassifier_32f::classifierStep`

#### 8.4.1.4 `Npp32s* NppiHaarClassifier_32f::counterDevice`

#### 8.4.1.5 `int NppiHaarClassifier_32f::numClassifiers`

number of classifiers

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

## 8.5 NppiHOGConfig Struct Reference

The [NppiHOGConfig](#) structure defines the configuration parameters for the HOG descriptor.:

```
#include <nppdefs.h>
```

### Data Fields

- [int cellSize](#)  
*square cell size (pixels).*
- [int histogramBlockSize](#)  
*square histogram block size (pixels).*
- [int nHistogramBins](#)  
*required number of histogram bins.*
- [NppiSize detectionWindowSize](#)  
*detection window size (pixels).*

### 8.5.1 Detailed Description

The [NppiHOGConfig](#) structure defines the configuration parameters for the HOG descriptor.:

### 8.5.2 Field Documentation

#### 8.5.2.1 int NppiHOGConfig::cellSize

square cell size (pixels).

#### 8.5.2.2 NppiSize NppiHOGConfig::detectionWindowSize

detection window size (pixels).

#### 8.5.2.3 int NppiHOGConfig::histogramBlockSize

square histogram block size (pixels).

#### 8.5.2.4 int NppiHOGConfig::nHistogramBins

required number of histogram bins.

The documentation for this struct was generated from the following file:

- C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h

## 8.6 NppiPoint Struct Reference

2D Point

```
#include <nppdefs.h>
```

### Data Fields

- `int x`  
*x-coordinate.*
- `int y`  
*y-coordinate.*

### 8.6.1 Detailed Description

2D Point

### 8.6.2 Field Documentation

#### 8.6.2.1 `int NppiPoint::x`

x-coordinate.

#### 8.6.2.2 `int NppiPoint::y`

y-coordinate.

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`



## 8.7 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 (lowest memory address).*
- `int y`  
*y-coordinate of upper left corner (lowest memory address).*
- `int width`  
*Rectangle width.*
- `int height`  
*Rectangle height.*

### 8.7.1 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.

### 8.7.2 Field Documentation

#### 8.7.2.1 `int NppiRect::height`

Rectangle height.

#### 8.7.2.2 `int NppiRect::width`

Rectangle width.

#### 8.7.2.3 `int NppiRect::x`

x-coordinate of upper left corner (lowest memory address).

#### 8.7.2.4 `int NppiRect::y`

y-coordinate of upper left corner (lowest memory address).

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

## 8.8 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.*

### 8.8.1 Detailed Description

2D Size This struct typically represents the size of a rectangular region in two space.

### 8.8.2 Field Documentation

#### 8.8.2.1 `int NppiSize::height`

Rectangle height.

#### 8.8.2.2 `int NppiSize::width`

Rectangle width.

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

## 8.9 NppLibraryVersion Struct Reference

```
#include <nppdefs.h>
```

### Data Fields

- int `major`  
*Major version number.*
- int `minor`  
*Minor version number.*
- int `build`  
*Build number.*

### 8.9.1 Field Documentation

#### 8.9.1.1 int NppLibraryVersion::build

Build number.

This reflects the nightly build this release was made from.

#### 8.9.1.2 int NppLibraryVersion::major

Major version number.

#### 8.9.1.3 int NppLibraryVersion::minor

Minor version number.

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

## 8.10 NppPointPolar Struct Reference

2D Polar Point

```
#include <nppdefs.h>
```

### Data Fields

- [Npp32f rho](#)
- [Npp32f theta](#)

### 8.10.1 Detailed Description

2D Polar Point

### 8.10.2 Field Documentation

#### 8.10.2.1 Npp32f NppPointPolar::rho

#### 8.10.2.2 Npp32f NppPointPolar::theta

The documentation for this struct was generated from the following file:

- `C:/src/sw/rel/gpgpu/toolkit/r9.0/NPP/npp/include/nppdefs.h`

# Index

- `__align__`
  - `npp_basic_types`, [49](#), [50](#)
- Basic NPP Data Types, [47](#)
- build
  - `NppLibraryVersion`, [173](#)
- cellSize
  - `NppiHOGConfig`, [169](#)
- classifiers
  - `NppiHaarClassifier_32f`, [168](#)
- classifierSize
  - `NppiHaarClassifier_32f`, [168](#)
- classifierStep
  - `NppiHaarClassifier_32f`, [168](#)
- Compare Operations, [141](#)
- core\_npp
  - `nppGetGpuComputeCapability`, [28](#)
  - `nppGetGpuDeviceProperties`, [28](#)
  - `nppGetGpuName`, [28](#)
  - `nppGetGpuNumSMs`, [28](#)
  - `nppGetLibVersion`, [28](#)
  - `nppGetMaxThreadsPerBlock`, [29](#)
  - `nppGetMaxThreadsPerSM`, [29](#)
  - `nppGetStream`, [29](#)
  - `nppGetStreamMaxThreadsPerSM`, [29](#)
  - `nppGetStreamNumSMs`, [29](#)
  - `nppSetStream`, [29](#)
- counterDevice
  - `NppiHaarClassifier_32f`, [168](#)
- detectionWindowSize
  - `NppiHOGConfig`, [169](#)
- haarBuffer
  - `NppiHaarBuffer`, [167](#)
- haarBufferSize
  - `NppiHaarBuffer`, [167](#)
- height
  - `NppiRect`, [171](#)
  - `NppiSize`, [172](#)
- histogramBlockSize
  - `NppiHOGConfig`, [169](#)
- im
  - `NPP_ALIGN_16`, [163](#)
  - `NPP_ALIGN_8`, [165](#)
- image\_compare\_operations
  - `nppiCompare_16s_AC4R`, [144](#)
  - `nppiCompare_16s_C1R`, [145](#)
  - `nppiCompare_16s_C3R`, [145](#)
  - `nppiCompare_16s_C4R`, [145](#)
  - `nppiCompare_16u_AC4R`, [146](#)
  - `nppiCompare_16u_C1R`, [146](#)
  - `nppiCompare_16u_C3R`, [147](#)
  - `nppiCompare_16u_C4R`, [147](#)
  - `nppiCompare_32f_AC4R`, [148](#)
  - `nppiCompare_32f_C1R`, [148](#)
  - `nppiCompare_32f_C3R`, [149](#)
  - `nppiCompare_32f_C4R`, [149](#)
  - `nppiCompare_8u_AC4R`, [150](#)
  - `nppiCompare_8u_C1R`, [150](#)
  - `nppiCompare_8u_C3R`, [151](#)
  - `nppiCompare_8u_C4R`, [151](#)
  - `nppiCompareC_16s_AC4R`, [152](#)
  - `nppiCompareC_16s_C1R`, [152](#)
  - `nppiCompareC_16s_C3R`, [153](#)
  - `nppiCompareC_16s_C4R`, [153](#)
  - `nppiCompareC_16u_AC4R`, [154](#)
  - `nppiCompareC_16u_C1R`, [154](#)
  - `nppiCompareC_16u_C3R`, [154](#)
  - `nppiCompareC_16u_C4R`, [155](#)
  - `nppiCompareC_32f_AC4R`, [155](#)
  - `nppiCompareC_32f_C1R`, [156](#)
  - `nppiCompareC_32f_C3R`, [156](#)
  - `nppiCompareC_32f_C4R`, [157](#)
  - `nppiCompareC_8u_AC4R`, [157](#)
  - `nppiCompareC_8u_C1R`, [157](#)
  - `nppiCompareC_8u_C3R`, [158](#)
  - `nppiCompareC_8u_C4R`, [158](#)
  - `nppiCompareEqualEps_32f_AC4R`, [159](#)
  - `nppiCompareEqualEps_32f_C1R`, [159](#)
  - `nppiCompareEqualEps_32f_C3R`, [160](#)
  - `nppiCompareEqualEps_32f_C4R`, [160](#)
  - `nppiCompareEqualEpsC_32f_AC4R`, [161](#)
  - `nppiCompareEqualEpsC_32f_C1R`, [161](#)
  - `nppiCompareEqualEpsC_32f_C3R`, [162](#)
  - `nppiCompareEqualEpsC_32f_C4R`, [162](#)
- image\_threshold\_operations
  - `nppiThreshold_16s_AC4IR`, [66](#)
  - `nppiThreshold_16s_AC4R`, [66](#)

- nppiThreshold\_16s\_C1IR, 67  
 nppiThreshold\_16s\_C1R, 67  
 nppiThreshold\_16s\_C3IR, 68  
 nppiThreshold\_16s\_C3R, 68  
 nppiThreshold\_16u\_AC4IR, 69  
 nppiThreshold\_16u\_AC4R, 69  
 nppiThreshold\_16u\_C1IR, 69  
 nppiThreshold\_16u\_C1R, 70  
 nppiThreshold\_16u\_C3IR, 70  
 nppiThreshold\_16u\_C3R, 71  
 nppiThreshold\_32f\_AC4IR, 71  
 nppiThreshold\_32f\_AC4R, 72  
 nppiThreshold\_32f\_C1IR, 72  
 nppiThreshold\_32f\_C1R, 73  
 nppiThreshold\_32f\_C3IR, 73  
 nppiThreshold\_32f\_C3R, 73  
 nppiThreshold\_8u\_AC4IR, 74  
 nppiThreshold\_8u\_AC4R, 74  
 nppiThreshold\_8u\_C1IR, 75  
 nppiThreshold\_8u\_C1R, 75  
 nppiThreshold\_8u\_C3IR, 76  
 nppiThreshold\_8u\_C3R, 76  
 nppiThreshold\_GT\_16s\_AC4IR, 77  
 nppiThreshold\_GT\_16s\_AC4R, 77  
 nppiThreshold\_GT\_16s\_C1IR, 78  
 nppiThreshold\_GT\_16s\_C1R, 78  
 nppiThreshold\_GT\_16s\_C3IR, 78  
 nppiThreshold\_GT\_16s\_C3R, 79  
 nppiThreshold\_GT\_16u\_AC4IR, 79  
 nppiThreshold\_GT\_16u\_AC4R, 80  
 nppiThreshold\_GT\_16u\_C1IR, 80  
 nppiThreshold\_GT\_16u\_C1R, 80  
 nppiThreshold\_GT\_16u\_C3IR, 81  
 nppiThreshold\_GT\_16u\_C3R, 81  
 nppiThreshold\_GT\_32f\_AC4IR, 82  
 nppiThreshold\_GT\_32f\_AC4R, 82  
 nppiThreshold\_GT\_32f\_C1IR, 82  
 nppiThreshold\_GT\_32f\_C1R, 83  
 nppiThreshold\_GT\_32f\_C3IR, 83  
 nppiThreshold\_GT\_32f\_C3R, 84  
 nppiThreshold\_GT\_8u\_AC4IR, 84  
 nppiThreshold\_GT\_8u\_AC4R, 84  
 nppiThreshold\_GT\_8u\_C1IR, 85  
 nppiThreshold\_GT\_8u\_C1R, 85  
 nppiThreshold\_GT\_8u\_C3IR, 86  
 nppiThreshold\_GT\_8u\_C3R, 86  
 nppiThreshold\_GTVAl\_16s\_AC4IR, 86  
 nppiThreshold\_GTVAl\_16s\_AC4R, 87  
 nppiThreshold\_GTVAl\_16s\_C1IR, 87  
 nppiThreshold\_GTVAl\_16s\_C1R, 88  
 nppiThreshold\_GTVAl\_16s\_C3IR, 88  
 nppiThreshold\_GTVAl\_16s\_C3R, 88  
 nppiThreshold\_GTVAl\_16u\_AC4IR, 89  
 nppiThreshold\_GTVAl\_16u\_AC4R, 89  
 nppiThreshold\_GTVAl\_16u\_C1IR, 90  
 nppiThreshold\_GTVAl\_16u\_C1R, 90  
 nppiThreshold\_GTVAl\_16u\_C3IR, 91  
 nppiThreshold\_GTVAl\_16u\_C3R, 91  
 nppiThreshold\_GTVAl\_32f\_AC4IR, 91  
 nppiThreshold\_GTVAl\_32f\_AC4R, 92  
 nppiThreshold\_GTVAl\_32f\_C1IR, 92  
 nppiThreshold\_GTVAl\_32f\_C1R, 93  
 nppiThreshold\_GTVAl\_32f\_C3IR, 93  
 nppiThreshold\_GTVAl\_32f\_C3R, 93  
 nppiThreshold\_GTVAl\_8u\_AC4IR, 94  
 nppiThreshold\_GTVAl\_8u\_AC4R, 94  
 nppiThreshold\_GTVAl\_8u\_C1IR, 95  
 nppiThreshold\_GTVAl\_8u\_C1R, 95  
 nppiThreshold\_GTVAl\_8u\_C3IR, 96  
 nppiThreshold\_GTVAl\_8u\_C3R, 96  
 nppiThreshold\_LT\_16s\_AC4IR, 96  
 nppiThreshold\_LT\_16s\_AC4R, 97  
 nppiThreshold\_LT\_16s\_C1IR, 97  
 nppiThreshold\_LT\_16s\_C1R, 98  
 nppiThreshold\_LT\_16s\_C3IR, 98  
 nppiThreshold\_LT\_16s\_C3R, 98  
 nppiThreshold\_LT\_16u\_AC4IR, 99  
 nppiThreshold\_LT\_16u\_AC4R, 99  
 nppiThreshold\_LT\_16u\_C1IR, 100  
 nppiThreshold\_LT\_16u\_C1R, 100  
 nppiThreshold\_LT\_16u\_C3IR, 100  
 nppiThreshold\_LT\_16u\_C3R, 101  
 nppiThreshold\_LT\_32f\_AC4IR, 101  
 nppiThreshold\_LT\_32f\_AC4R, 102  
 nppiThreshold\_LT\_32f\_C1IR, 102  
 nppiThreshold\_LT\_32f\_C1R, 102  
 nppiThreshold\_LT\_32f\_C3IR, 103  
 nppiThreshold\_LT\_32f\_C3R, 103  
 nppiThreshold\_LT\_8u\_AC4IR, 104  
 nppiThreshold\_LT\_8u\_AC4R, 104  
 nppiThreshold\_LT\_8u\_C1IR, 104  
 nppiThreshold\_LT\_8u\_C1R, 105  
 nppiThreshold\_LT\_8u\_C3IR, 105  
 nppiThreshold\_LT\_8u\_C3R, 106  
 nppiThreshold\_LTVAl\_16s\_AC4IR, 106  
 nppiThreshold\_LTVAl\_16s\_AC4R, 106  
 nppiThreshold\_LTVAl\_16s\_C1IR, 107  
 nppiThreshold\_LTVAl\_16s\_C1R, 107  
 nppiThreshold\_LTVAl\_16s\_C3IR, 108  
 nppiThreshold\_LTVAl\_16s\_C3R, 108  
 nppiThreshold\_LTVAl\_16u\_AC4IR, 109  
 nppiThreshold\_LTVAl\_16u\_AC4R, 109  
 nppiThreshold\_LTVAl\_16u\_C1IR, 109  
 nppiThreshold\_LTVAl\_16u\_C1R, 110  
 nppiThreshold\_LTVAl\_16u\_C3IR, 110  
 nppiThreshold\_LTVAl\_16u\_C3R, 111  
 nppiThreshold\_LTVAl\_32f\_AC4IR, 111  
 nppiThreshold\_LTVAl\_32f\_AC4R, 111

- nppiThreshold\_LTVal\_32f\_C1IR, 112
- nppiThreshold\_LTVal\_32f\_C1R, 112
- nppiThreshold\_LTVal\_32f\_C3IR, 113
- nppiThreshold\_LTVal\_32f\_C3R, 113
- nppiThreshold\_LTVal\_8u\_AC4IR, 114
- nppiThreshold\_LTVal\_8u\_AC4R, 114
- nppiThreshold\_LTVal\_8u\_C1IR, 114
- nppiThreshold\_LTVal\_8u\_C1R, 115
- nppiThreshold\_LTVal\_8u\_C3IR, 115
- nppiThreshold\_LTVal\_8u\_C3R, 116
- nppiThreshold\_LTValGTVal\_16s\_AC4IR, 116
- nppiThreshold\_LTValGTVal\_16s\_AC4R, 117
- nppiThreshold\_LTValGTVal\_16s\_C1IR, 117
- nppiThreshold\_LTValGTVal\_16s\_C1R, 118
- nppiThreshold\_LTValGTVal\_16s\_C3IR, 118
- nppiThreshold\_LTValGTVal\_16s\_C3R, 119
- nppiThreshold\_LTValGTVal\_16u\_AC4IR, 119
- nppiThreshold\_LTValGTVal\_16u\_AC4R, 120
- nppiThreshold\_LTValGTVal\_16u\_C1IR, 120
- nppiThreshold\_LTValGTVal\_16u\_C1R, 121
- nppiThreshold\_LTValGTVal\_16u\_C3IR, 121
- nppiThreshold\_LTValGTVal\_16u\_C3R, 122
- nppiThreshold\_LTValGTVal\_32f\_AC4IR, 122
- nppiThreshold\_LTValGTVal\_32f\_AC4R, 123
- nppiThreshold\_LTValGTVal\_32f\_C1IR, 123
- nppiThreshold\_LTValGTVal\_32f\_C1R, 124
- nppiThreshold\_LTValGTVal\_32f\_C3IR, 124
- nppiThreshold\_LTValGTVal\_32f\_C3R, 125
- nppiThreshold\_LTValGTVal\_8u\_AC4IR, 125
- nppiThreshold\_LTValGTVal\_8u\_AC4R, 126
- nppiThreshold\_LTValGTVal\_8u\_C1IR, 126
- nppiThreshold\_LTValGTVal\_8u\_C1R, 127
- nppiThreshold\_LTValGTVal\_8u\_C3IR, 127
- nppiThreshold\_LTValGTVal\_8u\_C3R, 128
- nppiThreshold\_Val\_16s\_AC4IR, 128
- nppiThreshold\_Val\_16s\_AC4R, 129
- nppiThreshold\_Val\_16s\_C1IR, 129
- nppiThreshold\_Val\_16s\_C1R, 130
- nppiThreshold\_Val\_16s\_C3IR, 130
- nppiThreshold\_Val\_16s\_C3R, 131
- nppiThreshold\_Val\_16u\_AC4IR, 131
- nppiThreshold\_Val\_16u\_AC4R, 132
- nppiThreshold\_Val\_16u\_C1IR, 132
- nppiThreshold\_Val\_16u\_C1R, 133
- nppiThreshold\_Val\_16u\_C3IR, 133
- nppiThreshold\_Val\_16u\_C3R, 134
- nppiThreshold\_Val\_32f\_AC4IR, 134
- nppiThreshold\_Val\_32f\_AC4R, 135
- nppiThreshold\_Val\_32f\_C1IR, 135
- nppiThreshold\_Val\_32f\_C1R, 136
- nppiThreshold\_Val\_32f\_C3IR, 136
- nppiThreshold\_Val\_32f\_C3R, 137
- nppiThreshold\_Val\_8u\_AC4IR, 137
- nppiThreshold\_Val\_8u\_AC4R, 138
- nppiThreshold\_Val\_8u\_C1IR, 138
- nppiThreshold\_Val\_8u\_C1R, 139
- nppiThreshold\_Val\_8u\_C3IR, 139
- nppiThreshold\_Val\_8u\_C3R, 140
- major
  - NppLibraryVersion, 173
- minor
  - NppLibraryVersion, 173
- nHistogramBins
  - NppiHOGConfig, 169
- NPP Core, 27
- NPP Type Definitions and Constants, 31
- Npp16s
  - npp\_basic\_types, 48
- Npp16sc
  - npp\_basic\_types, 50
- Npp16u
  - npp\_basic\_types, 48
- Npp16uc
  - npp\_basic\_types, 50
- Npp32f
  - npp\_basic\_types, 48
- Npp32fc
  - npp\_basic\_types, 48
- Npp32s
  - npp\_basic\_types, 48
- Npp32sc
  - npp\_basic\_types, 48
- Npp32u
  - npp\_basic\_types, 49
- Npp32uc
  - npp\_basic\_types, 49
- Npp64f
  - npp\_basic\_types, 49
- Npp64fc
  - npp\_basic\_types, 49
- Npp64s
  - npp\_basic\_types, 49
- Npp64sc
  - npp\_basic\_types, 49
- Npp64u
  - npp\_basic\_types, 49
- Npp8s
  - npp\_basic\_types, 49
- Npp8u
  - npp\_basic\_types, 49
- Npp8uc
  - npp\_basic\_types, 50
- NPP\_AFFINE\_QUAD\_INCORRECT\_WARNING
  - typedefs\_npp, 46
- NPP\_ALG\_HINT\_ACCURATE

- typedefs\_npp, 41
- NPP\_ALG\_HINT\_FAST
  - typedefs\_npp, 41
- NPP\_ALG\_HINT\_NONE
  - typedefs\_npp, 41
- NPP\_ALIGNMENT\_ERROR
  - typedefs\_npp, 44
- NPP\_ANCHOR\_ERROR
  - typedefs\_npp, 45
- NPP\_BAD\_ARGUMENT\_ERROR
  - typedefs\_npp, 45
- NPP\_BORDER\_CONSTANT
  - typedefs\_npp, 42
- NPP\_BORDER\_MIRROR
  - typedefs\_npp, 42
- NPP\_BORDER\_NONE
  - typedefs\_npp, 42
- NPP\_BORDER\_REPLICATE
  - typedefs\_npp, 42
- NPP\_BORDER\_UNDEFINED
  - typedefs\_npp, 42
- NPP\_BORDER\_WRAP
  - typedefs\_npp, 42
- NPP\_BOTH\_AXIS
  - typedefs\_npp, 41
- NPP\_CHANNEL\_ERROR
  - typedefs\_npp, 45
- NPP\_CHANNEL\_ORDER\_ERROR
  - typedefs\_npp, 45
- NPP\_CMP\_EQ
  - typedefs\_npp, 40
- NPP\_CMP\_GREATER
  - typedefs\_npp, 40
- NPP\_CMP\_GREATER\_EQ
  - typedefs\_npp, 40
- NPP\_CMP\_LESS
  - typedefs\_npp, 40
- NPP\_CMP\_LESS\_EQ
  - typedefs\_npp, 40
- NPP\_COEFFICIENT\_ERROR
  - typedefs\_npp, 45
- NPP\_COI\_ERROR
  - typedefs\_npp, 45
- NPP\_CONTEXT\_MATCH\_ERROR
  - typedefs\_npp, 45
- NPP\_CORRUPTED\_DATA\_ERROR
  - typedefs\_npp, 45
- NPP\_CUDA\_1\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_1\_1
  - typedefs\_npp, 40
- NPP\_CUDA\_1\_2
  - typedefs\_npp, 40
- NPP\_CUDA\_1\_3
  - typedefs\_npp, 40
- NPP\_CUDA\_2\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_2\_1
  - typedefs\_npp, 40
- NPP\_CUDA\_3\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_3\_2
  - typedefs\_npp, 40
- NPP\_CUDA\_3\_5
  - typedefs\_npp, 40
- NPP\_CUDA\_3\_7
  - typedefs\_npp, 40
- NPP\_CUDA\_5\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_5\_2
  - typedefs\_npp, 40
- NPP\_CUDA\_5\_3
  - typedefs\_npp, 40
- NPP\_CUDA\_6\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_6\_1
  - typedefs\_npp, 40
- NPP\_CUDA\_6\_2
  - typedefs\_npp, 40
- NPP\_CUDA\_6\_3
  - typedefs\_npp, 40
- NPP\_CUDA\_7\_0
  - typedefs\_npp, 40
- NPP\_CUDA\_KERNEL\_EXECUTION\_ERROR
  - typedefs\_npp, 44
- NPP\_CUDA\_NOT\_CAPABLE
  - typedefs\_npp, 40
- NPP\_CUDA\_UNKNOWN\_VERSION
  - typedefs\_npp, 40
- NPP\_DATA\_TYPE\_ERROR
  - typedefs\_npp, 45
- NPP\_DIVIDE\_BY\_ZERO\_ERROR
  - typedefs\_npp, 45
- NPP\_DIVIDE\_BY\_ZERO\_WARNING
  - typedefs\_npp, 46
- NPP\_DIVISOR\_ERROR
  - typedefs\_npp, 45
- NPP\_DOUBLE\_SIZE\_WARNING
  - typedefs\_npp, 46
- NPP\_ERROR
  - typedefs\_npp, 45
- NPP\_ERROR\_RESERVED
  - typedefs\_npp, 45
- NPP\_FFT\_FLAG\_ERROR
  - typedefs\_npp, 45
- NPP\_FFT\_ORDER\_ERROR
  - typedefs\_npp, 45
- NPP\_FILTER\_SCHARR
  - typedefs\_npp, 40



- typedefs\_npp, 42
- NPP\_FILTER\_SOBEL
  - typedefs\_npp, 42
- NPP\_HAAR\_CLASSIFIER\_PIXEL\_MATCH\_-
  - ERROR
  - typedefs\_npp, 44
- NPP\_HISTOGRAM\_NUMBER\_OF\_LEVELS\_-
  - ERROR
  - typedefs\_npp, 44
- NPP\_HORIZONTAL\_AXIS
  - typedefs\_npp, 41
- NPP\_INTERPOLATION\_ERROR
  - typedefs\_npp, 45
- NPP\_INVALID\_DEVICE\_POINTER\_ERROR
  - typedefs\_npp, 44
- NPP\_INVALID\_HOST\_POINTER\_ERROR
  - typedefs\_npp, 44
- NPP\_LUT\_NUMBER\_OF\_LEVELS\_ERROR
  - typedefs\_npp, 45
- NPP\_LUT\_PALETTE\_BITSIZE\_ERROR
  - typedefs\_npp, 44
- NPP\_MASK\_SIZE\_11\_X\_11
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_13\_X\_13
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_15\_X\_15
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_1\_X\_3
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_1\_X\_5
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_3\_X\_1
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_3\_X\_3
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_5\_X\_1
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_5\_X\_5
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_7\_X\_7
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_9\_X\_9
  - typedefs\_npp, 43
- NPP\_MASK\_SIZE\_ERROR
  - typedefs\_npp, 45
- NPP\_MEMCPY\_ERROR
  - typedefs\_npp, 44
- NPP\_MEMFREE\_ERROR
  - typedefs\_npp, 44
- NPP\_MEMORY\_ALLOCATION\_ERR
  - typedefs\_npp, 45
- NPP\_MEMSET\_ERROR
  - typedefs\_npp, 44
- NPP\_MIRROR\_FLIP\_ERROR
  - typedefs\_npp, 45
- NPP\_MISALIGNED\_DST\_ROI\_WARNING
  - typedefs\_npp, 46
- NPP\_MOMENT\_00\_ZERO\_ERROR
  - typedefs\_npp, 45
- NPP\_NO\_ERROR
  - typedefs\_npp, 45
- NPP\_NO\_MEMORY\_ERROR
  - typedefs\_npp, 45
- NPP\_NO\_OPERATION\_WARNING
  - typedefs\_npp, 45
- NPP\_NOT\_EVEN\_STEP\_ERROR
  - typedefs\_npp, 44
- NPP\_NOT\_IMPLEMENTED\_ERROR
  - typedefs\_npp, 45
- NPP\_NOT\_SUFFICIENT\_COMPUTE\_-
  - CAPABILITY
  - typedefs\_npp, 44
- NPP\_NOT\_SUPPORTED\_MODE\_ERROR
  - typedefs\_npp, 44
- NPP\_NULL\_POINTER\_ERROR
  - typedefs\_npp, 45
- NPP\_NUMBER\_OF\_CHANNELS\_ERROR
  - typedefs\_npp, 45
- NPP\_OUT\_OFF\_RANGE\_ERROR
  - typedefs\_npp, 45
- NPP\_OVERFLOW\_ERROR
  - typedefs\_npp, 44
- NPP\_QUADRANGLE\_ERROR
  - typedefs\_npp, 45
- NPP\_QUALITY\_INDEX\_ERROR
  - typedefs\_npp, 44
- NPP\_RANGE\_ERROR
  - typedefs\_npp, 45
- NPP\_RECTANGLE\_ERROR
  - typedefs\_npp, 45
- NPP\_RESIZE\_FACTOR\_ERROR
  - typedefs\_npp, 45
- NPP\_RESIZE\_NO\_OPERATION\_ERROR
  - typedefs\_npp, 44
- NPP\_RND\_FINANCIAL
  - typedefs\_npp, 43
- NPP\_RND\_NEAR
  - typedefs\_npp, 43
- NPP\_RND\_ZERO
  - typedefs\_npp, 44
- NPP\_ROUND\_MODE\_NOT\_SUPPORTED\_-
  - ERROR
  - typedefs\_npp, 44
- NPP\_ROUND\_NEAREST\_TIES\_AWAY\_-
  - FROM\_ZERO
  - typedefs\_npp, 44
- NPP\_ROUND\_NEAREST\_TIES\_TO\_EVEN
  - typedefs\_npp, 43

- NPP\_ROUND\_TOWARD\_ZERO
  - typedefs\_npp, 44
- NPP\_SCALE\_RANGE\_ERROR
  - typedefs\_npp, 45
- NPP\_SIZE\_ERROR
  - typedefs\_npp, 45
- NPP\_STEP\_ERROR
  - typedefs\_npp, 45
- NPP\_STRIDE\_ERROR
  - typedefs\_npp, 45
- NPP\_SUCCESS
  - typedefs\_npp, 45
- NPP\_TEXTURE\_BIND\_ERROR
  - typedefs\_npp, 44
- NPP\_THRESHOLD\_ERROR
  - typedefs\_npp, 45
- NPP\_THRESHOLD\_NEGATIVE\_LEVEL\_ -
  - ERROR
  - typedefs\_npp, 45
- NPP\_VERTICAL\_AXIS
  - typedefs\_npp, 41
- NPP\_WRONG\_INTERSECTION\_QUAD\_ -
  - WARNING
  - typedefs\_npp, 46
- NPP\_WRONG\_INTERSECTION\_ROI\_ERROR
  - typedefs\_npp, 44
- NPP\_WRONG\_INTERSECTION\_ROI\_ -
  - WARNING
  - typedefs\_npp, 46
- NPP\_ZC\_MODE\_NOT\_SUPPORTED\_ERROR
  - typedefs\_npp, 44
- NPP\_ZERO\_MASK\_VALUE\_ERROR
  - typedefs\_npp, 45
- NPP\_ALIGN\_16, 163
  - im, 163
  - re, 164
- NPP\_ALIGN\_8, 165
  - im, 165
  - re, 165, 166
- npp\_basic\_types
  - \_\_align\_\_, 49, 50
  - Npp16s, 48
  - Npp16sc, 50
  - Npp16u, 48
  - Npp16uc, 50
  - Npp32f, 48
  - Npp32fc, 48
  - Npp32s, 48
  - Npp32sc, 48
  - Npp32u, 49
  - Npp32uc, 49
  - Npp64f, 49
  - Npp64fc, 49
  - Npp64s, 49
  - Npp64sc, 49
  - Npp64u, 49
  - Npp8s, 49
  - Npp8u, 49
  - Npp8uc, 50
- NPP\_HOG\_MAX\_BINS\_PER\_CELL
  - typedefs\_npp, 37
- NPP\_HOG\_MAX\_BLOCK\_SIZE
  - typedefs\_npp, 37
- NPP\_HOG\_MAX\_CELL\_SIZE
  - typedefs\_npp, 37
- NPP\_HOG\_MAX\_CELLS\_PER\_DESCRIPTOR
  - typedefs\_npp, 37
- NPP\_HOG\_MAX\_DESCRIPTOR\_ -
  - LOCATIONS\_PER\_CALL
  - typedefs\_npp, 38
- NPP\_HOG\_MAX\_OVERLAPPING\_BLOCKS\_ -
  - PER\_DESCRIPTOR
  - typedefs\_npp, 38
- NPP\_MAX\_16S
  - typedefs\_npp, 38
- NPP\_MAX\_16U
  - typedefs\_npp, 38
- NPP\_MAX\_32S
  - typedefs\_npp, 38
- NPP\_MAX\_32U
  - typedefs\_npp, 38
- NPP\_MAX\_64S
  - typedefs\_npp, 38
- NPP\_MAX\_64U
  - typedefs\_npp, 38
- NPP\_MAX\_8S
  - typedefs\_npp, 38
- NPP\_MAX\_8U
  - typedefs\_npp, 38
- NPP\_MAXABS\_32F
  - typedefs\_npp, 38
- NPP\_MAXABS\_64F
  - typedefs\_npp, 39
- NPP\_MIN\_16S
  - typedefs\_npp, 39
- NPP\_MIN\_16U
  - typedefs\_npp, 39
- NPP\_MIN\_32S
  - typedefs\_npp, 39
- NPP\_MIN\_32U
  - typedefs\_npp, 39
- NPP\_MIN\_64S
  - typedefs\_npp, 39
- NPP\_MIN\_64U
  - typedefs\_npp, 39
- NPP\_MIN\_8S
  - typedefs\_npp, 39
- NPP\_MIN\_8U
  - typedefs\_npp, 39

- typedefs\_npp, 39
- NPP\_MINABS\_32F
  - typedefs\_npp, 39
- NPP\_MINABS\_64F
  - typedefs\_npp, 39
- NppCmpOp
  - typedefs\_npp, 40
- nppGetGpuComputeCapability
  - core\_npp, 28
- nppGetGpuDeviceProperties
  - core\_npp, 28
- nppGetGpuName
  - core\_npp, 28
- nppGetGpuNumSMs
  - core\_npp, 28
- nppGetLibVersion
  - core\_npp, 28
- nppGetMaxThreadsPerBlock
  - core\_npp, 29
- nppGetMaxThreadsPerSM
  - core\_npp, 29
- nppGetStream
  - core\_npp, 29
- nppGetStreamMaxThreadsPerSM
  - core\_npp, 29
- nppGetStreamNumSMs
  - core\_npp, 29
- NppGpuComputeCapability
  - typedefs\_npp, 40
- NppHintAlgorithm
  - typedefs\_npp, 40
- NPPI\_BAYER\_BGGR
  - typedefs\_npp, 41
- NPPI\_BAYER\_GBRG
  - typedefs\_npp, 41
- NPPI\_BAYER\_GRBG
  - typedefs\_npp, 41
- NPPI\_BAYER\_RGGB
  - typedefs\_npp, 41
- NPPI\_INTER\_CUBIC
  - typedefs\_npp, 42
- NPPI\_INTER\_CUBIC2P\_B05C03
  - typedefs\_npp, 42
- NPPI\_INTER\_CUBIC2P\_BSPLINE
  - typedefs\_npp, 42
- NPPI\_INTER\_CUBIC2P\_CATMULLROM
  - typedefs\_npp, 42
- NPPI\_INTER\_LANCZOS
  - typedefs\_npp, 42
- NPPI\_INTER\_LANCZOS3\_ADVANCED
  - typedefs\_npp, 42
- NPPI\_INTER\_LINEAR
  - typedefs\_npp, 42
- NPPI\_INTER\_NN
  - typedefs\_npp, 42
- NPPI\_INTER\_SUPER
  - typedefs\_npp, 42
- NPPI\_INTER\_UNDEFINED
  - typedefs\_npp, 42
- NPPI\_OP\_ALPHA\_ATOP
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_ATOP\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_IN
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_IN\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_OUT
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_OUT\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_OVER
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_OVER\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_PLUS
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_PLUS\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_PREMUL
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_XOR
  - typedefs\_npp, 41
- NPPI\_OP\_ALPHA\_XOR\_PREMUL
  - typedefs\_npp, 41
- NPPI\_SMOOTH\_EDGE
  - typedefs\_npp, 42
- nppiACTable
  - typedefs\_npp, 42
- NppiAlphaOp
  - typedefs\_npp, 41
- NppiAxis
  - typedefs\_npp, 41
- NppiBayerGridPosition
  - typedefs\_npp, 41
- NppiBorderType
  - typedefs\_npp, 41
- nppiCompare\_16s\_AC4R
  - image\_compare\_operations, 144
- nppiCompare\_16s\_C1R
  - image\_compare\_operations, 145
- nppiCompare\_16s\_C3R
  - image\_compare\_operations, 145
- nppiCompare\_16s\_C4R
  - image\_compare\_operations, 145
- nppiCompare\_16u\_AC4R
  - image\_compare\_operations, 146
- nppiCompare\_16u\_C1R

- image\_compare\_operations, 146
- nppiCompare\_16u\_C3R
  - image\_compare\_operations, 147
- nppiCompare\_16u\_C4R
  - image\_compare\_operations, 147
- nppiCompare\_32f\_AC4R
  - image\_compare\_operations, 148
- nppiCompare\_32f\_C1R
  - image\_compare\_operations, 148
- nppiCompare\_32f\_C3R
  - image\_compare\_operations, 149
- nppiCompare\_32f\_C4R
  - image\_compare\_operations, 149
- nppiCompare\_8u\_AC4R
  - image\_compare\_operations, 150
- nppiCompare\_8u\_C1R
  - image\_compare\_operations, 150
- nppiCompare\_8u\_C3R
  - image\_compare\_operations, 151
- nppiCompare\_8u\_C4R
  - image\_compare\_operations, 151
- nppiCompareC\_16s\_AC4R
  - image\_compare\_operations, 152
- nppiCompareC\_16s\_C1R
  - image\_compare\_operations, 152
- nppiCompareC\_16s\_C3R
  - image\_compare\_operations, 153
- nppiCompareC\_16s\_C4R
  - image\_compare\_operations, 153
- nppiCompareC\_16u\_AC4R
  - image\_compare\_operations, 154
- nppiCompareC\_16u\_C1R
  - image\_compare\_operations, 154
- nppiCompareC\_16u\_C3R
  - image\_compare\_operations, 154
- nppiCompareC\_16u\_C4R
  - image\_compare\_operations, 155
- nppiCompareC\_32f\_AC4R
  - image\_compare\_operations, 155
- nppiCompareC\_32f\_C1R
  - image\_compare\_operations, 156
- nppiCompareC\_32f\_C3R
  - image\_compare\_operations, 156
- nppiCompareC\_32f\_C4R
  - image\_compare\_operations, 157
- nppiCompareC\_8u\_AC4R
  - image\_compare\_operations, 157
- nppiCompareC\_8u\_C1R
  - image\_compare\_operations, 157
- nppiCompareC\_8u\_C3R
  - image\_compare\_operations, 158
- nppiCompareC\_8u\_C4R
  - image\_compare\_operations, 158
- nppiCompareEqualEps\_32f\_AC4R
  - image\_compare\_operations, 159
- nppiCompareEqualEps\_32f\_C1R
  - image\_compare\_operations, 159
- nppiCompareEqualEps\_32f\_C3R
  - image\_compare\_operations, 160
- nppiCompareEqualEps\_32f\_C4R
  - image\_compare\_operations, 160
- nppiCompareEqualEpsC\_32f\_AC4R
  - image\_compare\_operations, 161
- nppiCompareEqualEpsC\_32f\_C1R
  - image\_compare\_operations, 161
- nppiCompareEqualEpsC\_32f\_C3R
  - image\_compare\_operations, 162
- nppiCompareEqualEpsC\_32f\_C4R
  - image\_compare\_operations, 162
- nppiDCTable
  - typedefs\_npp, 42
- NppiDifferentialKernel
  - typedefs\_npp, 42
- NppiHaarBuffer, 167
  - haarBuffer, 167
  - haarBufferSize, 167
- NppiHaarClassifier\_32f, 168
  - classifiers, 168
  - classifierSize, 168
  - classifierStep, 168
  - counterDevice, 168
  - numClassifiers, 168
- NppiHOGConfig, 169
  - cellSize, 169
  - detectionWindowSize, 169
  - histogramBlockSize, 169
  - nHistogramBins, 169
- NppiHuffmanTableType
  - typedefs\_npp, 42
- NppiInterpolationMode
  - typedefs\_npp, 42
- NppiMaskSize
  - typedefs\_npp, 42
- NppiNorm
  - typedefs\_npp, 43
- nppiNormInf
  - typedefs\_npp, 43
- nppiNormL1
  - typedefs\_npp, 43
- nppiNormL2
  - typedefs\_npp, 43
- NppiPoint, 170
  - x, 170
  - y, 170
- NppiRect, 171
  - height, 171
  - width, 171
  - x, 171

- y, 171
- NppiSize, 172
  - height, 172
  - width, 172
- nppiThreshold\_16s\_AC4IR
  - image\_threshold\_operations, 66
- nppiThreshold\_16s\_AC4R
  - image\_threshold\_operations, 66
- nppiThreshold\_16s\_C1IR
  - image\_threshold\_operations, 67
- nppiThreshold\_16s\_C1R
  - image\_threshold\_operations, 67
- nppiThreshold\_16s\_C3IR
  - image\_threshold\_operations, 68
- nppiThreshold\_16s\_C3R
  - image\_threshold\_operations, 68
- nppiThreshold\_16u\_AC4IR
  - image\_threshold\_operations, 69
- nppiThreshold\_16u\_AC4R
  - image\_threshold\_operations, 69
- nppiThreshold\_16u\_C1IR
  - image\_threshold\_operations, 69
- nppiThreshold\_16u\_C1R
  - image\_threshold\_operations, 70
- nppiThreshold\_16u\_C3IR
  - image\_threshold\_operations, 70
- nppiThreshold\_16u\_C3R
  - image\_threshold\_operations, 71
- nppiThreshold\_32f\_AC4IR
  - image\_threshold\_operations, 71
- nppiThreshold\_32f\_AC4R
  - image\_threshold\_operations, 72
- nppiThreshold\_32f\_C1IR
  - image\_threshold\_operations, 72
- nppiThreshold\_32f\_C1R
  - image\_threshold\_operations, 73
- nppiThreshold\_32f\_C3IR
  - image\_threshold\_operations, 73
- nppiThreshold\_32f\_C3R
  - image\_threshold\_operations, 73
- nppiThreshold\_8u\_AC4IR
  - image\_threshold\_operations, 74
- nppiThreshold\_8u\_AC4R
  - image\_threshold\_operations, 74
- nppiThreshold\_8u\_C1IR
  - image\_threshold\_operations, 75
- nppiThreshold\_8u\_C1R
  - image\_threshold\_operations, 75
- nppiThreshold\_8u\_C3IR
  - image\_threshold\_operations, 76
- nppiThreshold\_8u\_C3R
  - image\_threshold\_operations, 76
- nppiThreshold\_GT\_16s\_AC4IR
  - image\_threshold\_operations, 77
- nppiThreshold\_GT\_16s\_AC4R
  - image\_threshold\_operations, 77
- nppiThreshold\_GT\_16s\_C1IR
  - image\_threshold\_operations, 78
- nppiThreshold\_GT\_16s\_C1R
  - image\_threshold\_operations, 78
- nppiThreshold\_GT\_16s\_C3IR
  - image\_threshold\_operations, 78
- nppiThreshold\_GT\_16s\_C3R
  - image\_threshold\_operations, 79
- nppiThreshold\_GT\_16u\_AC4IR
  - image\_threshold\_operations, 79
- nppiThreshold\_GT\_16u\_AC4R
  - image\_threshold\_operations, 80
- nppiThreshold\_GT\_16u\_C1IR
  - image\_threshold\_operations, 80
- nppiThreshold\_GT\_16u\_C1R
  - image\_threshold\_operations, 80
- nppiThreshold\_GT\_16u\_C3IR
  - image\_threshold\_operations, 81
- nppiThreshold\_GT\_16u\_C3R
  - image\_threshold\_operations, 81
- nppiThreshold\_GT\_32f\_AC4IR
  - image\_threshold\_operations, 82
- nppiThreshold\_GT\_32f\_AC4R
  - image\_threshold\_operations, 82
- nppiThreshold\_GT\_32f\_C1IR
  - image\_threshold\_operations, 82
- nppiThreshold\_GT\_32f\_C1R
  - image\_threshold\_operations, 83
- nppiThreshold\_GT\_32f\_C3IR
  - image\_threshold\_operations, 83
- nppiThreshold\_GT\_32f\_C3R
  - image\_threshold\_operations, 84
- nppiThreshold\_GT\_8u\_AC4IR
  - image\_threshold\_operations, 84
- nppiThreshold\_GT\_8u\_AC4R
  - image\_threshold\_operations, 84
- nppiThreshold\_GT\_8u\_C1IR
  - image\_threshold\_operations, 85
- nppiThreshold\_GT\_8u\_C1R
  - image\_threshold\_operations, 85
- nppiThreshold\_GT\_8u\_C3IR
  - image\_threshold\_operations, 86
- nppiThreshold\_GT\_8u\_C3R
  - image\_threshold\_operations, 86
- nppiThreshold\_GTVal\_16s\_AC4IR
  - image\_threshold\_operations, 86
- nppiThreshold\_GTVal\_16s\_AC4R
  - image\_threshold\_operations, 87
- nppiThreshold\_GTVal\_16s\_C1IR
  - image\_threshold\_operations, 87
- nppiThreshold\_GTVal\_16s\_C1R
  - image\_threshold\_operations, 88

- [nppiThreshold\\_GTVal\\_16s\\_C3IR](#)  
[image\\_threshold\\_operations, 88](#)
- [nppiThreshold\\_GTVal\\_16s\\_C3R](#)  
[image\\_threshold\\_operations, 88](#)
- [nppiThreshold\\_GTVal\\_16u\\_AC4IR](#)  
[image\\_threshold\\_operations, 89](#)
- [nppiThreshold\\_GTVal\\_16u\\_AC4R](#)  
[image\\_threshold\\_operations, 89](#)
- [nppiThreshold\\_GTVal\\_16u\\_C1IR](#)  
[image\\_threshold\\_operations, 90](#)
- [nppiThreshold\\_GTVal\\_16u\\_C1R](#)  
[image\\_threshold\\_operations, 90](#)
- [nppiThreshold\\_GTVal\\_16u\\_C3IR](#)  
[image\\_threshold\\_operations, 91](#)
- [nppiThreshold\\_GTVal\\_16u\\_C3R](#)  
[image\\_threshold\\_operations, 91](#)
- [nppiThreshold\\_GTVal\\_32f\\_AC4IR](#)  
[image\\_threshold\\_operations, 91](#)
- [nppiThreshold\\_GTVal\\_32f\\_AC4R](#)  
[image\\_threshold\\_operations, 92](#)
- [nppiThreshold\\_GTVal\\_32f\\_C1IR](#)  
[image\\_threshold\\_operations, 92](#)
- [nppiThreshold\\_GTVal\\_32f\\_C1R](#)  
[image\\_threshold\\_operations, 93](#)
- [nppiThreshold\\_GTVal\\_32f\\_C3IR](#)  
[image\\_threshold\\_operations, 93](#)
- [nppiThreshold\\_GTVal\\_32f\\_C3R](#)  
[image\\_threshold\\_operations, 93](#)
- [nppiThreshold\\_GTVal\\_8u\\_AC4IR](#)  
[image\\_threshold\\_operations, 94](#)
- [nppiThreshold\\_GTVal\\_8u\\_AC4R](#)  
[image\\_threshold\\_operations, 94](#)
- [nppiThreshold\\_GTVal\\_8u\\_C1IR](#)  
[image\\_threshold\\_operations, 95](#)
- [nppiThreshold\\_GTVal\\_8u\\_C1R](#)  
[image\\_threshold\\_operations, 95](#)
- [nppiThreshold\\_GTVal\\_8u\\_C3IR](#)  
[image\\_threshold\\_operations, 96](#)
- [nppiThreshold\\_GTVal\\_8u\\_C3R](#)  
[image\\_threshold\\_operations, 96](#)
- [nppiThreshold\\_LT\\_16s\\_AC4IR](#)  
[image\\_threshold\\_operations, 96](#)
- [nppiThreshold\\_LT\\_16s\\_AC4R](#)  
[image\\_threshold\\_operations, 97](#)
- [nppiThreshold\\_LT\\_16s\\_C1IR](#)  
[image\\_threshold\\_operations, 97](#)
- [nppiThreshold\\_LT\\_16s\\_C1R](#)  
[image\\_threshold\\_operations, 98](#)
- [nppiThreshold\\_LT\\_16s\\_C3IR](#)  
[image\\_threshold\\_operations, 98](#)
- [nppiThreshold\\_LT\\_16s\\_C3R](#)  
[image\\_threshold\\_operations, 98](#)
- [nppiThreshold\\_LT\\_16u\\_AC4IR](#)  
[image\\_threshold\\_operations, 99](#)
- [nppiThreshold\\_LT\\_16u\\_AC4R](#)  
[image\\_threshold\\_operations, 99](#)
- [nppiThreshold\\_LT\\_16u\\_C1IR](#)  
[image\\_threshold\\_operations, 100](#)
- [nppiThreshold\\_LT\\_16u\\_C1R](#)  
[image\\_threshold\\_operations, 100](#)
- [nppiThreshold\\_LT\\_16u\\_C3IR](#)  
[image\\_threshold\\_operations, 100](#)
- [nppiThreshold\\_LT\\_16u\\_C3R](#)  
[image\\_threshold\\_operations, 101](#)
- [nppiThreshold\\_LT\\_32f\\_AC4IR](#)  
[image\\_threshold\\_operations, 101](#)
- [nppiThreshold\\_LT\\_32f\\_AC4R](#)  
[image\\_threshold\\_operations, 102](#)
- [nppiThreshold\\_LT\\_32f\\_C1IR](#)  
[image\\_threshold\\_operations, 102](#)
- [nppiThreshold\\_LT\\_32f\\_C1R](#)  
[image\\_threshold\\_operations, 102](#)
- [nppiThreshold\\_LT\\_32f\\_C3IR](#)  
[image\\_threshold\\_operations, 103](#)
- [nppiThreshold\\_LT\\_32f\\_C3R](#)  
[image\\_threshold\\_operations, 103](#)
- [nppiThreshold\\_LT\\_8u\\_AC4IR](#)  
[image\\_threshold\\_operations, 104](#)
- [nppiThreshold\\_LT\\_8u\\_AC4R](#)  
[image\\_threshold\\_operations, 104](#)
- [nppiThreshold\\_LT\\_8u\\_C1IR](#)  
[image\\_threshold\\_operations, 104](#)
- [nppiThreshold\\_LT\\_8u\\_C1R](#)  
[image\\_threshold\\_operations, 105](#)
- [nppiThreshold\\_LT\\_8u\\_C3IR](#)  
[image\\_threshold\\_operations, 105](#)
- [nppiThreshold\\_LT\\_8u\\_C3R](#)  
[image\\_threshold\\_operations, 106](#)
- [nppiThreshold\\_LTVal\\_16s\\_AC4IR](#)  
[image\\_threshold\\_operations, 106](#)
- [nppiThreshold\\_LTVal\\_16s\\_AC4R](#)  
[image\\_threshold\\_operations, 106](#)
- [nppiThreshold\\_LTVal\\_16s\\_C1IR](#)  
[image\\_threshold\\_operations, 107](#)
- [nppiThreshold\\_LTVal\\_16s\\_C1R](#)  
[image\\_threshold\\_operations, 107](#)
- [nppiThreshold\\_LTVal\\_16s\\_C3IR](#)  
[image\\_threshold\\_operations, 108](#)
- [nppiThreshold\\_LTVal\\_16s\\_C3R](#)  
[image\\_threshold\\_operations, 108](#)
- [nppiThreshold\\_LTVal\\_16u\\_AC4IR](#)  
[image\\_threshold\\_operations, 109](#)
- [nppiThreshold\\_LTVal\\_16u\\_AC4R](#)  
[image\\_threshold\\_operations, 109](#)
- [nppiThreshold\\_LTVal\\_16u\\_C1IR](#)  
[image\\_threshold\\_operations, 109](#)
- [nppiThreshold\\_LTVal\\_16u\\_C1R](#)  
[image\\_threshold\\_operations, 110](#)



- nppiThreshold\_LTVal\_16u\_C3IR
  - image\_threshold\_operations, [110](#)
- nppiThreshold\_LTVal\_16u\_C3R
  - image\_threshold\_operations, [111](#)
- nppiThreshold\_LTVal\_32f\_AC4IR
  - image\_threshold\_operations, [111](#)
- nppiThreshold\_LTVal\_32f\_AC4R
  - image\_threshold\_operations, [111](#)
- nppiThreshold\_LTVal\_32f\_C1IR
  - image\_threshold\_operations, [112](#)
- nppiThreshold\_LTVal\_32f\_C1R
  - image\_threshold\_operations, [112](#)
- nppiThreshold\_LTVal\_32f\_C3IR
  - image\_threshold\_operations, [113](#)
- nppiThreshold\_LTVal\_32f\_C3R
  - image\_threshold\_operations, [113](#)
- nppiThreshold\_LTVal\_8u\_AC4IR
  - image\_threshold\_operations, [114](#)
- nppiThreshold\_LTVal\_8u\_AC4R
  - image\_threshold\_operations, [114](#)
- nppiThreshold\_LTVal\_8u\_C1IR
  - image\_threshold\_operations, [114](#)
- nppiThreshold\_LTVal\_8u\_C1R
  - image\_threshold\_operations, [115](#)
- nppiThreshold\_LTVal\_8u\_C3IR
  - image\_threshold\_operations, [115](#)
- nppiThreshold\_LTVal\_8u\_C3R
  - image\_threshold\_operations, [116](#)
- nppiThreshold\_LTValGTVal\_16s\_AC4IR
  - image\_threshold\_operations, [116](#)
- nppiThreshold\_LTValGTVal\_16s\_AC4R
  - image\_threshold\_operations, [117](#)
- nppiThreshold\_LTValGTVal\_16s\_C1IR
  - image\_threshold\_operations, [117](#)
- nppiThreshold\_LTValGTVal\_16s\_C1R
  - image\_threshold\_operations, [118](#)
- nppiThreshold\_LTValGTVal\_16s\_C3IR
  - image\_threshold\_operations, [118](#)
- nppiThreshold\_LTValGTVal\_16s\_C3R
  - image\_threshold\_operations, [119](#)
- nppiThreshold\_LTValGTVal\_16u\_AC4IR
  - image\_threshold\_operations, [119](#)
- nppiThreshold\_LTValGTVal\_16u\_AC4R
  - image\_threshold\_operations, [120](#)
- nppiThreshold\_LTValGTVal\_16u\_C1IR
  - image\_threshold\_operations, [120](#)
- nppiThreshold\_LTValGTVal\_16u\_C1R
  - image\_threshold\_operations, [121](#)
- nppiThreshold\_LTValGTVal\_16u\_C3IR
  - image\_threshold\_operations, [121](#)
- nppiThreshold\_LTValGTVal\_16u\_C3R
  - image\_threshold\_operations, [122](#)
- nppiThreshold\_LTValGTVal\_32f\_AC4IR
  - image\_threshold\_operations, [122](#)
- nppiThreshold\_LTValGTVal\_32f\_AC4R
  - image\_threshold\_operations, [123](#)
- nppiThreshold\_LTValGTVal\_32f\_C1IR
  - image\_threshold\_operations, [123](#)
- nppiThreshold\_LTValGTVal\_32f\_C1R
  - image\_threshold\_operations, [124](#)
- nppiThreshold\_LTValGTVal\_32f\_C3IR
  - image\_threshold\_operations, [124](#)
- nppiThreshold\_LTValGTVal\_32f\_C3R
  - image\_threshold\_operations, [125](#)
- nppiThreshold\_LTValGTVal\_8u\_AC4IR
  - image\_threshold\_operations, [125](#)
- nppiThreshold\_LTValGTVal\_8u\_AC4R
  - image\_threshold\_operations, [126](#)
- nppiThreshold\_LTValGTVal\_8u\_C1IR
  - image\_threshold\_operations, [126](#)
- nppiThreshold\_LTValGTVal\_8u\_C1R
  - image\_threshold\_operations, [127](#)
- nppiThreshold\_LTValGTVal\_8u\_C3IR
  - image\_threshold\_operations, [127](#)
- nppiThreshold\_LTValGTVal\_8u\_C3R
  - image\_threshold\_operations, [128](#)
- nppiThreshold\_Val\_16s\_AC4IR
  - image\_threshold\_operations, [128](#)
- nppiThreshold\_Val\_16s\_AC4R
  - image\_threshold\_operations, [129](#)
- nppiThreshold\_Val\_16s\_C1IR
  - image\_threshold\_operations, [129](#)
- nppiThreshold\_Val\_16s\_C1R
  - image\_threshold\_operations, [130](#)
- nppiThreshold\_Val\_16s\_C3IR
  - image\_threshold\_operations, [130](#)
- nppiThreshold\_Val\_16s\_C3R
  - image\_threshold\_operations, [131](#)
- nppiThreshold\_Val\_16u\_AC4IR
  - image\_threshold\_operations, [131](#)
- nppiThreshold\_Val\_16u\_AC4R
  - image\_threshold\_operations, [132](#)
- nppiThreshold\_Val\_16u\_C1IR
  - image\_threshold\_operations, [132](#)
- nppiThreshold\_Val\_16u\_C1R
  - image\_threshold\_operations, [133](#)
- nppiThreshold\_Val\_16u\_C3IR
  - image\_threshold\_operations, [133](#)
- nppiThreshold\_Val\_16u\_C3R
  - image\_threshold\_operations, [134](#)
- nppiThreshold\_Val\_32f\_AC4IR
  - image\_threshold\_operations, [134](#)
- nppiThreshold\_Val\_32f\_AC4R
  - image\_threshold\_operations, [135](#)
- nppiThreshold\_Val\_32f\_C1IR
  - image\_threshold\_operations, [135](#)
- nppiThreshold\_Val\_32f\_C1R
  - image\_threshold\_operations, [136](#)

- nppiThreshold\_Val\_32f\_C3IR
  - image\_threshold\_operations, 136
- nppiThreshold\_Val\_32f\_C3R
  - image\_threshold\_operations, 137
- nppiThreshold\_Val\_8u\_AC4IR
  - image\_threshold\_operations, 137
- nppiThreshold\_Val\_8u\_AC4R
  - image\_threshold\_operations, 138
- nppiThreshold\_Val\_8u\_C1IR
  - image\_threshold\_operations, 138
- nppiThreshold\_Val\_8u\_C1R
  - image\_threshold\_operations, 139
- nppiThreshold\_Val\_8u\_C3IR
  - image\_threshold\_operations, 139
- nppiThreshold\_Val\_8u\_C3R
  - image\_threshold\_operations, 140
- NppLibraryVersion, 173
  - build, 173
  - major, 173
  - minor, 173
- NppPointPolar, 174
  - rho, 174
  - theta, 174
- NppRoundMode
  - typedefs\_npp, 43
- nppSetStream
  - core\_npp, 29
- NppStatus
  - typedefs\_npp, 44
- NppsZCType
  - typedefs\_npp, 46
- nppZCC
  - typedefs\_npp, 46
- nppZCR
  - typedefs\_npp, 46
- nppZCXor
  - typedefs\_npp, 46
- numClassifiers
  - NppiHaarClassifier\_32f, 168
- re
  - NPP\_ALIGN\_16, 164
  - NPP\_ALIGN\_8, 165, 166
- rho
  - NppPointPolar, 174
- theta
  - NppPointPolar, 174
- Threshold and Compare Operations, 51
- Threshold Operations, 52
- typedefs\_npp
  - NPP\_AFFINE\_QUAD\_INCORRECT\_WARNING, 46
  - NPP\_ALG\_HINT\_ACCURATE, 41
  - NPP\_ALG\_HINT\_FAST, 41
  - NPP\_ALG\_HINT\_NONE, 41
  - NPP\_ALIGNMENT\_ERROR, 44
  - NPP\_ANCHOR\_ERROR, 45
  - NPP\_BAD\_ARGUMENT\_ERROR, 45
  - NPP\_BORDER\_CONSTANT, 42
  - NPP\_BORDER\_MIRROR, 42
  - NPP\_BORDER\_NONE, 42
  - NPP\_BORDER\_REPLICATE, 42
  - NPP\_BORDER\_UNDEFINED, 42
  - NPP\_BORDER\_WRAP, 42
  - NPP\_BOTH\_AXIS, 41
  - NPP\_CHANNEL\_ERROR, 45
  - NPP\_CHANNEL\_ORDER\_ERROR, 45
  - NPP\_CMP\_EQ, 40
  - NPP\_CMP\_GREATER, 40
  - NPP\_CMP\_GREATER\_EQ, 40
  - NPP\_CMP\_LESS, 40
  - NPP\_CMP\_LESS\_EQ, 40
  - NPP\_COEFFICIENT\_ERROR, 45
  - NPP\_COI\_ERROR, 45
  - NPP\_CONTEXT\_MATCH\_ERROR, 45
  - NPP\_CORRUPTED\_DATA\_ERROR, 45
  - NPP\_CUDA\_1\_0, 40
  - NPP\_CUDA\_1\_1, 40
  - NPP\_CUDA\_1\_2, 40
  - NPP\_CUDA\_1\_3, 40
  - NPP\_CUDA\_2\_0, 40
  - NPP\_CUDA\_2\_1, 40
  - NPP\_CUDA\_3\_0, 40
  - NPP\_CUDA\_3\_2, 40
  - NPP\_CUDA\_3\_5, 40
  - NPP\_CUDA\_3\_7, 40
  - NPP\_CUDA\_5\_0, 40
  - NPP\_CUDA\_5\_2, 40
  - NPP\_CUDA\_5\_3, 40
  - NPP\_CUDA\_6\_0, 40
  - NPP\_CUDA\_6\_1, 40
  - NPP\_CUDA\_6\_2, 40
  - NPP\_CUDA\_6\_3, 40
  - NPP\_CUDA\_7\_0, 40
  - NPP\_CUDA\_KERNEL\_EXECUTION\_ERROR, 44
  - NPP\_CUDA\_NOT\_CAPABLE, 40
  - NPP\_CUDA\_UNKNOWN\_VERSION, 40
  - NPP\_DATA\_TYPE\_ERROR, 45
  - NPP\_DIVIDE\_BY\_ZERO\_ERROR, 45
  - NPP\_DIVIDE\_BY\_ZERO\_WARNING, 46
  - NPP\_DIVISOR\_ERROR, 45
  - NPP\_DOUBLE\_SIZE\_WARNING, 46
  - NPP\_ERROR, 45
  - NPP\_ERROR\_RESERVED, 45
  - NPP\_FFT\_FLAG\_ERROR, 45
  - NPP\_FFT\_ORDER\_ERROR, 45



- NPP\_FILTER\_SCHARR, 42  
 NPP\_FILTER\_SOBEL, 42  
 NPP\_HAAR\_CLASSIFIER\_PIXEL\_-  
   MATCH\_ERROR, 44  
 NPP\_HISTOGRAM\_NUMBER\_OF\_-  
   LEVELS\_ERROR, 44  
 NPP\_HORIZONTAL\_AXIS, 41  
 NPP\_INTERPOLATION\_ERROR, 45  
 NPP\_INVALID\_DEVICE\_POINTER\_-  
   ERROR, 44  
 NPP\_INVALID\_HOST\_POINTER\_ERROR,  
   44  
 NPP\_LUT\_NUMBER\_OF\_LEVELS\_-  
   ERROR, 45  
 NPP\_LUT\_PALETTE\_BITSIZE\_ERROR, 44  
 NPP\_MASK\_SIZE\_11\_X\_11, 43  
 NPP\_MASK\_SIZE\_13\_X\_13, 43  
 NPP\_MASK\_SIZE\_15\_X\_15, 43  
 NPP\_MASK\_SIZE\_1\_X\_3, 43  
 NPP\_MASK\_SIZE\_1\_X\_5, 43  
 NPP\_MASK\_SIZE\_3\_X\_1, 43  
 NPP\_MASK\_SIZE\_3\_X\_3, 43  
 NPP\_MASK\_SIZE\_5\_X\_1, 43  
 NPP\_MASK\_SIZE\_5\_X\_5, 43  
 NPP\_MASK\_SIZE\_7\_X\_7, 43  
 NPP\_MASK\_SIZE\_9\_X\_9, 43  
 NPP\_MASK\_SIZE\_ERROR, 45  
 NPP\_MEMCPY\_ERROR, 44  
 NPP\_MEMFREE\_ERROR, 44  
 NPP\_MEMORY\_ALLOCATION\_ERR, 45  
 NPP\_MEMSET\_ERROR, 44  
 NPP\_MIRROR\_FLIP\_ERROR, 45  
 NPP\_MISALIGNED\_DST\_ROI\_WARNING,  
   46  
 NPP\_MOMENT\_00\_ZERO\_ERROR, 45  
 NPP\_NO\_ERROR, 45  
 NPP\_NO\_MEMORY\_ERROR, 45  
 NPP\_NO\_OPERATION\_WARNING, 45  
 NPP\_NOT\_EVEN\_STEP\_ERROR, 44  
 NPP\_NOT\_IMPLEMENTED\_ERROR, 45  
 NPP\_NOT\_SUFFICIENT\_COMPUTE\_-  
   CAPABILITY, 44  
 NPP\_NOT\_SUPPORTED\_MODE\_ERROR,  
   44  
 NPP\_NULL\_POINTER\_ERROR, 45  
 NPP\_NUMBER\_OF\_CHANNELS\_ERROR,  
   45  
 NPP\_OUT\_OFF\_RANGE\_ERROR, 45  
 NPP\_OVERFLOW\_ERROR, 44  
 NPP\_QUADRANGLE\_ERROR, 45  
 NPP\_QUALITY\_INDEX\_ERROR, 44  
 NPP\_RANGE\_ERROR, 45  
 NPP\_RECTANGLE\_ERROR, 45  
 NPP\_RESIZE\_FACTOR\_ERROR, 45  
 NPP\_RESIZE\_NO\_OPERATION\_ERROR,  
   44  
 NPP\_RND\_FINANCIAL, 43  
 NPP\_RND\_NEAR, 43  
 NPP\_RND\_ZERO, 44  
 NPP\_ROUND\_MODE\_NOT\_-  
   SUPPORTED\_ERROR, 44  
 NPP\_ROUND\_NEAREST\_TIES\_AWAY\_-  
   FROM\_ZERO, 44  
 NPP\_ROUND\_NEAREST\_TIES\_TO\_EVEN,  
   43  
 NPP\_ROUND\_TOWARD\_ZERO, 44  
 NPP\_SCALE\_RANGE\_ERROR, 45  
 NPP\_SIZE\_ERROR, 45  
 NPP\_STEP\_ERROR, 45  
 NPP\_STRIDE\_ERROR, 45  
 NPP\_SUCCESS, 45  
 NPP\_TEXTURE\_BIND\_ERROR, 44  
 NPP\_THRESHOLD\_ERROR, 45  
 NPP\_THRESHOLD\_NEGATIVE\_LEVEL\_-  
   ERROR, 45  
 NPP\_VERTICAL\_AXIS, 41  
 NPP\_WRONG\_INTERSECTION\_QUAD\_-  
   WARNING, 46  
 NPP\_WRONG\_INTERSECTION\_ROI\_-  
   ERROR, 44  
 NPP\_WRONG\_INTERSECTION\_ROI\_-  
   WARNING, 46  
 NPP\_ZC\_MODE\_NOT\_SUPPORTED\_-  
   ERROR, 44  
 NPP\_ZERO\_MASK\_VALUE\_ERROR, 45  
 NPPI\_BAYER\_BGGR, 41  
 NPPI\_BAYER\_GBRG, 41  
 NPPI\_BAYER\_GRGB, 41  
 NPPI\_BAYER\_RGGB, 41  
 NPPI\_INTER\_CUBIC, 42  
 NPPI\_INTER\_CUBIC2P\_B05C03, 42  
 NPPI\_INTER\_CUBIC2P\_BSPLINE, 42  
 NPPI\_INTER\_CUBIC2P\_CATMULLROM,  
   42  
 NPPI\_INTER\_LANCZOS, 42  
 NPPI\_INTER\_LANCZOS3\_ADVANCED, 42  
 NPPI\_INTER\_LINEAR, 42  
 NPPI\_INTER\_NN, 42  
 NPPI\_INTER\_SUPER, 42  
 NPPI\_INTER\_UNDEFINED, 42  
 NPPI\_OP\_ALPHA\_ATOP, 41  
 NPPI\_OP\_ALPHA\_ATOP\_PREMUL, 41  
 NPPI\_OP\_ALPHA\_IN, 41  
 NPPI\_OP\_ALPHA\_IN\_PREMUL, 41  
 NPPI\_OP\_ALPHA\_OUT, 41  
 NPPI\_OP\_ALPHA\_OUT\_PREMUL, 41  
 NPPI\_OP\_ALPHA\_OVER, 41  
 NPPI\_OP\_ALPHA\_OVER\_PREMUL, 41

- NPPI\_OP\_ALPHA\_PLUS, 41
- NPPI\_OP\_ALPHA\_PLUS\_PREMUL, 41
- NPPI\_OP\_ALPHA\_PREMUL, 41
- NPPI\_OP\_ALPHA\_XOR, 41
- NPPI\_OP\_ALPHA\_XOR\_PREMUL, 41
- NPPI\_SMOOTH\_EDGE, 42
- nppiACTable, 42
- nppiDCTable, 42
- nppiNormInf, 43
- nppiNormL1, 43
- nppiNormL2, 43
- nppZCC, 46
- nppZCR, 46
- nppZCXor, 46
- typedefs\_npp
  - NPP\_HOG\_MAX\_BINS\_PER\_CELL, 37
  - NPP\_HOG\_MAX\_BLOCK\_SIZE, 37
  - NPP\_HOG\_MAX\_CELL\_SIZE, 37
  - NPP\_HOG\_MAX\_CELLS\_PER\_DESCRIPTOR, 37
  - NPP\_HOG\_MAX\_DESCRIPTOR\_LOCATIONS\_PER\_CALL, 38
  - NPP\_HOG\_MAX\_OVERLAPPING\_BLOCKS\_PER\_DESCRIPTOR, 38
  - NPP\_MAX\_16S, 38
  - NPP\_MAX\_16U, 38
  - NPP\_MAX\_32S, 38
  - NPP\_MAX\_32U, 38
  - NPP\_MAX\_64S, 38
  - NPP\_MAX\_64U, 38
  - NPP\_MAX\_8S, 38
  - NPP\_MAX\_8U, 38
  - NPP\_MAXABS\_32F, 38
  - NPP\_MAXABS\_64F, 39
  - NPP\_MIN\_16S, 39
  - NPP\_MIN\_16U, 39
  - NPP\_MIN\_32S, 39
  - NPP\_MIN\_32U, 39
  - NPP\_MIN\_64S, 39
  - NPP\_MIN\_64U, 39
  - NPP\_MIN\_8S, 39
  - NPP\_MIN\_8U, 39
  - NPP\_MINABS\_32F, 39
  - NPP\_MINABS\_64F, 39
  - NppCmpOp, 40
  - NppGpuComputeCapability, 40
  - NppHintAlgorithm, 40
  - NppiAlphaOp, 41
  - NppiAxis, 41
  - NppiBayerGridPosition, 41
  - NppiBorderType, 41
  - NppiDifferentialKernel, 42
  - NppiHuffmanTableType, 42
  - NppiInterpolationMode, 42
  - NppiMaskSize, 42
  - NppiNorm, 43
  - NppRoundMode, 43
  - NppStatus, 44
  - NppsZCType, 46
  - width
    - NppiRect, 171
    - NppiSize, 172
  - x
    - NppiPoint, 170
    - NppiRect, 171
  - y
    - NppiPoint, 170
    - NppiRect, 171