



# mambaApi Reference Manual

Automatically generated using doxygen

## Contents

<b>1</b>	<b>Data Structures</b>	<b>18</b>
<b>2</b>	<b>File List</b>	<b>18</b>
<b>3</b>	<b>MB_Basins_Ctx Struct Reference</b>	<b>20</b>
3.1	Detailed Description . . . . .	21
3.2	Field Documentation . . . . .	21
3.2.1	bytes_marker . . . . .	21
3.2.2	current_water_level . . . . .	21
3.2.3	height . . . . .	21
3.2.4	HierarchicalList . . . . .	21
3.2.5	InsertNeighbors . . . . .	21
3.2.6	linoff_marker . . . . .	21
3.2.7	linoff_src . . . . .	21
3.2.8	plines_marker . . . . .	21
3.2.9	plines_src . . . . .	21
3.2.10	TokensArray . . . . .	21
3.2.11	width . . . . .	22
<b>4</b>	<b>MB_Hierarbld_Ctx Struct Reference</b>	<b>22</b>
4.1	Detailed Description . . . . .	22
4.2	Field Documentation . . . . .	22
4.2.1	bytes . . . . .	22
4.2.2	current_water_level . . . . .	22
4.2.3	height . . . . .	22
4.2.4	HierarchicalList . . . . .	22
4.2.5	InsertNeighbors . . . . .	22
4.2.6	linoff_mask . . . . .	23
4.2.7	linoff_srcdest . . . . .	23
4.2.8	pix_status . . . . .	23
4.2.9	plines_mask . . . . .	23
4.2.10	plines_srcdest . . . . .	23
4.2.11	TokensArray . . . . .	23
4.2.12	width . . . . .	23
<b>5</b>	<b>MB_Hierardualbld_Ctx Struct Reference</b>	<b>23</b>
5.1	Detailed Description . . . . .	23
5.2	Field Documentation . . . . .	24
5.2.1	bytes . . . . .	24
5.2.2	current_water_level . . . . .	24

5.2.3	height	24
5.2.4	HierarchicalList	24
5.2.5	InsertNeighbors	24
5.2.6	linoff_mask	24
5.2.7	linoff_srcestd	24
5.2.8	pix_status	24
5.2.9	plines_mask	24
5.2.10	plines_srcestd	24
5.2.11	TokensArray	24
5.2.12	width	24
<b>6</b>	<b>MB_Image Struct Reference</b>	<b>25</b>
6.1	Detailed Description	25
6.2	Field Documentation	25
6.2.1	depth	25
6.2.2	height	25
6.2.3	PIXARRAY	25
6.2.4	PLINES	25
6.2.5	width	25
<b>7</b>	<b>MB_Label Struct Reference</b>	<b>25</b>
7.1	Detailed Description	26
7.2	Field Documentation	26
7.2.1	ccurrent	26
7.2.2	CEQ	26
7.2.3	current	26
7.2.4	EQ	26
7.2.5	maxEQ	26
<b>8</b>	<b>MB_ListControl Struct Reference</b>	<b>26</b>
8.1	Detailed Description	26
8.2	Field Documentation	27
8.2.1	firstx	27
8.2.2	firsty	27
8.2.3	lastx	27
8.2.4	lasty	27
<b>9</b>	<b>MB-Token Struct Reference</b>	<b>27</b>
9.1	Detailed Description	27
9.2	Field Documentation	27
9.2.1	nextx	27

9.2.2	nexty	27
<b>10</b>	<b>MB_Watershed_Ctx Struct Reference</b>	<b>28</b>
10.1	Detailed Description	28
10.2	Field Documentation	28
10.2.1	bytes_marker	28
10.2.2	current_water_level	28
10.2.3	height	28
10.2.4	HierarchicalList	28
10.2.5	InsertNeighbors	28
10.2.6	linoff_marker	28
10.2.7	linoff_src	28
10.2.8	plines_marker	29
10.2.9	plines_src	29
10.2.10	TokensArray	29
10.2.11	toreinsertList	29
10.2.12	width	29
<b>11</b>	<b>mambaApi.h File Reference</b>	<b>29</b>
11.1	Detailed Description	31
11.2	Function Documentation	31
11.2.1	MB_Add	31
11.2.2	MB_And	32
11.2.3	MB_Basins	32
11.2.4	MB_BinHitOrMiss	32
11.2.5	MB_BldNb32	33
11.2.6	MB_BldNb8	33
11.2.7	MB_BldNbb	33
11.2.8	MB_Check	34
11.2.9	MB_Compare	34
11.2.10	MB_ConAdd	34
11.2.11	MB_ConDiv	35
11.2.12	MB_ConMul	35
11.2.13	MB_ConSet	35
11.2.14	MB_ConSub	35
11.2.15	MB_Convert	36
11.2.16	MB_Copy	36
11.2.17	MB_CopyBitPlane	36
11.2.18	MB_CopyBytePlane	36
11.2.19	MB_CopyLine	37
11.2.20	MB_Create	37

11.2.21 MB_CropCopy	37
11.2.22 MB_depthRange	38
11.2.23 MB_Destroy	38
11.2.24 MB_Diff	38
11.2.25 MB_DiffNb8	38
11.2.26 MB_DiffNbb	39
11.2.27 MB_Distanceb	39
11.2.28 MB_DualBldNb32	39
11.2.29 MB_DualBldNb8	40
11.2.30 MB_DualBldNbb	40
11.2.31 MB_Extract	40
11.2.32 MB_Frame	40
11.2.33 MB_GetPixel	41
11.2.34 MB_HierarBld	41
11.2.35 MB_HierarDualBld	41
11.2.36 MB_Histo	42
11.2.37 MB_Inf	42
11.2.38 MB_InfFarNb32	42
11.2.39 MB_InfFarNb8	42
11.2.40 MB_InfFarNbb	43
11.2.41 MB_InfNb32	43
11.2.42 MB_InfNb8	43
11.2.43 MB_InfNbb	44
11.2.44 MB_Inv	44
11.2.45 MB_Labelb	44
11.2.46 MB_Load	45
11.2.47 MB_Lookup	45
11.2.48 MB_Mask	45
11.2.49 MB_Mul	46
11.2.50 MB_Or	46
11.2.51 MB_PutPixel	46
11.2.52 MB_Range	46
11.2.53 MB_Shift32	47
11.2.54 MB_Shift8	47
11.2.55 MB_Shiftb	47
11.2.56 MB_Sub	48
11.2.57 MB_Sup	48
11.2.58 MB_SupFarNb32	48
11.2.59 MB_SupFarNb8	49
11.2.60 MB_SupFarNbb	49

11.2.61 MB_SupMask . . . . .	49
11.2.62 MB_SupNb32 . . . . .	50
11.2.63 MB_SupNb8 . . . . .	50
11.2.64 MB_SupNbb . . . . .	50
11.2.65 MB_Thresh . . . . .	51
11.2.66 MB_Volume . . . . .	51
11.2.67 MB_Watershed . . . . .	51
11.2.68 MB_Xor . . . . .	51
11.3 Variable Documentation . . . . .	52
11.3.1 MB_refcounter . . . . .	52
<b>12 mambaApi_loc.h File Reference</b>	<b>52</b>
12.1 Detailed Description . . . . .	53
12.2 Define Documentation . . . . .	53
12.2.1 MB_CHECK_SIZE_2 . . . . .	53
12.2.2 MB_CHECK_SIZE_3 . . . . .	53
12.2.3 MB_LIST_END . . . . .	53
12.2.4 MB_PROBE_PAIR . . . . .	53
12.3 Function Documentation . . . . .	53
12.3.1 MB_aligned_free . . . . .	53
12.3.2 MB_aligned_malloc . . . . .	53
12.3.3 MB_free . . . . .	54
12.3.4 MB_malloc . . . . .	54
12.3.5 MB_memcpy . . . . .	54
12.3.6 MB_memset . . . . .	54
12.4 Variable Documentation . . . . .	54
12.4.1 hxNbDir . . . . .	54
12.4.2 MB_VolumePerByte . . . . .	54
12.4.3 sqNbDir . . . . .	55
<b>13 mambaCommon.h File Reference</b>	<b>55</b>
13.1 Detailed Description . . . . .	55
13.2 Define Documentation . . . . .	56
13.2.1 BIN_FILL_VALUE . . . . .	56
13.2.2 GREY_FILL_VALUE . . . . .	56
13.2.3 I32_FILL_VALUE . . . . .	56
13.2.4 MB_LINE_COUNT . . . . .	56
13.2.5 MB_LINE_OFFSET . . . . .	56
13.2.6 MB_X_LEFT . . . . .	56
13.2.7 MB_X_RIGHT . . . . .	56
13.2.8 MB_Y_BOTTOM . . . . .	57

13.2.9	MB_Y_TOP	57
<b>13.3</b>	<b>Typedef Documentation</b>	<b>57</b>
13.3.1	PIX32	57
13.3.2	PIX8	57
13.3.3	PLINE	57
13.3.4	PLINE32	57
13.3.5	Sint16	57
13.3.6	Sint32	57
13.3.7	Sint64	57
13.3.8	Sint8	57
13.3.9	Uint16	57
13.3.10	Uint32	57
13.3.11	Uint64	58
13.3.12	Uint8	58
<b>13.4</b>	<b>Enumeration Type Documentation</b>	<b>58</b>
13.4.1	MB_edgemode_t	58
13.4.2	MB_grid_t	58
<b>14</b>	<b>MB_Add.c File Reference</b>	<b>58</b>
14.1	Detailed Description	58
14.2	Function Documentation	58
14.2.1	MB_Add	58
<b>15</b>	<b>MB_And.c File Reference</b>	<b>59</b>
15.1	Detailed Description	59
15.2	Function Documentation	59
15.2.1	MB_And	59
<b>16</b>	<b>MB_Basins.c File Reference</b>	<b>59</b>
16.1	Detailed Description	60
16.2	Typedef Documentation	60
16.2.1	TSWITCHEP	60
16.3	Function Documentation	60
16.3.1	MB_Basins	60
<b>17</b>	<b>MB_BinHitOrMiss.c File Reference</b>	<b>60</b>
17.1	Detailed Description	60
17.2	Function Documentation	61
17.2.1	MB_BinHitOrMiss	61
<b>18</b>	<b>MB_BldDirection.c File Reference</b>	<b>61</b>
18.1	Detailed Description	61

18.2	Typedef Documentation . . . . .	61
18.2.1	TSWITCHEP . . . . .	61
<b>19</b>	<b>MB_BldNb32.c File Reference</b>	<b>61</b>
19.1	Detailed Description . . . . .	62
19.2	Function Documentation . . . . .	62
19.2.1	MB_BldNb32 . . . . .	62
<b>20</b>	<b>MB_BldNb8.c File Reference</b>	<b>62</b>
20.1	Detailed Description . . . . .	62
20.2	Function Documentation . . . . .	63
20.2.1	MB_BldNb8 . . . . .	63
<b>21</b>	<b>MB_BldNbb.c File Reference</b>	<b>63</b>
21.1	Detailed Description . . . . .	63
21.2	Function Documentation . . . . .	63
21.2.1	MB_BldNbb . . . . .	63
<b>22</b>	<b>MB_Check.c File Reference</b>	<b>64</b>
22.1	Detailed Description . . . . .	64
22.2	Function Documentation . . . . .	64
22.2.1	MB_Check . . . . .	64
<b>23</b>	<b>MB_Compare.c File Reference</b>	<b>64</b>
23.1	Detailed Description . . . . .	64
23.2	Function Documentation . . . . .	65
23.2.1	MB_Compare . . . . .	65
<b>24</b>	<b>MB_ConAdd.c File Reference</b>	<b>65</b>
24.1	Detailed Description . . . . .	65
24.2	Function Documentation . . . . .	65
24.2.1	MB_ConAdd . . . . .	65
<b>25</b>	<b>MB_ConDiv.c File Reference</b>	<b>65</b>
25.1	Detailed Description . . . . .	66
25.2	Function Documentation . . . . .	66
25.2.1	MB_ConDiv . . . . .	66
<b>26</b>	<b>MB_ConMul.c File Reference</b>	<b>66</b>
26.1	Detailed Description . . . . .	66
26.2	Function Documentation . . . . .	66
26.2.1	MB_ConMul . . . . .	66
<b>27</b>	<b>MB_ConSet.c File Reference</b>	<b>67</b>

27.1 Detailed Description . . . . .	67
27.2 Function Documentation . . . . .	67
27.2.1 MB_ConSet . . . . .	67
<b>28 MB_ConSub.c File Reference</b>	<b>67</b>
28.1 Detailed Description . . . . .	67
28.2 Function Documentation . . . . .	68
28.2.1 MB_ConSub . . . . .	68
<b>29 MB_Convert.c File Reference</b>	<b>68</b>
29.1 Detailed Description . . . . .	68
29.2 Function Documentation . . . . .	68
29.2.1 MB_Convert . . . . .	68
29.2.2 MB_Convert1to8 . . . . .	69
29.2.3 MB_Convert8to1 . . . . .	69
<b>30 MB_Copy.c File Reference</b>	<b>69</b>
30.1 Detailed Description . . . . .	69
30.2 Function Documentation . . . . .	69
30.2.1 MB_Copy . . . . .	69
30.2.2 MB_CopyLine . . . . .	70
30.2.3 MB_CropCopy . . . . .	70
<b>31 MB_CopyBitPlane.c File Reference</b>	<b>70</b>
31.1 Detailed Description . . . . .	71
31.2 Function Documentation . . . . .	71
31.2.1 EXTRACT_BIT_PLANE . . . . .	71
31.2.2 MB_CopyBitPlane . . . . .	71
31.2.3 SET_BIT_PLANE . . . . .	71
<b>32 MB_CopyBytePlane.c File Reference</b>	<b>71</b>
32.1 Detailed Description . . . . .	72
32.2 Function Documentation . . . . .	72
32.2.1 MB_CopyBytePlane . . . . .	72
<b>33 MB_Create.c File Reference</b>	<b>72</b>
33.1 Detailed Description . . . . .	73
33.2 Define Documentation . . . . .	73
33.2.1 MB_MAX_IMAGE_H_SIZE . . . . .	73
33.2.2 MB_MAX_IMAGE_W_SIZE . . . . .	73
33.2.3 MB_ROUND_H . . . . .	73
33.2.4 MB_ROUND_W . . . . .	73

33.3	Function Documentation . . . . .	73
33.3.1	MB_Create . . . . .	73
33.3.2	MB_Destroy . . . . .	73
33.4	Variable Documentation . . . . .	74
33.4.1	MB_refcounter . . . . .	74
<b>34</b>	<b>MB_Diff.c File Reference</b>	<b>74</b>
34.1	Detailed Description . . . . .	74
34.2	Function Documentation . . . . .	74
34.2.1	MB_Diff . . . . .	74
<b>35</b>	<b>MB_DiffNb8.c File Reference</b>	<b>74</b>
35.1	Detailed Description . . . . .	75
35.2	Define Documentation . . . . .	75
35.2.1	EDGE_TYPE . . . . .	75
35.3	Function Documentation . . . . .	75
35.3.1	MB_DiffNb8 . . . . .	75
<b>36</b>	<b>MB_DiffNb8.c File Reference</b>	<b>75</b>
36.1	Detailed Description . . . . .	76
36.2	Define Documentation . . . . .	76
36.2.1	EDGE_TYPE . . . . .	76
36.3	Function Documentation . . . . .	76
36.3.1	MB_DiffNb8 . . . . .	76
<b>37</b>	<b>MB_Distanceb.c File Reference</b>	<b>76</b>
37.1	Detailed Description . . . . .	77
37.2	Define Documentation . . . . .	77
37.2.1	EDGE_DIST . . . . .	77
37.2.2	IS_SET . . . . .	77
37.2.3	LEFT . . . . .	77
37.2.4	RIGHT . . . . .	77
37.2.5	VAL . . . . .	77
37.3	Typedef Documentation . . . . .	77
37.3.1	TSWITCHEP . . . . .	77
37.4	Function Documentation . . . . .	77
37.4.1	MB_Distanceb . . . . .	77
<b>38</b>	<b>MB_DualBldNb32.c File Reference</b>	<b>78</b>
38.1	Detailed Description . . . . .	78
38.2	Function Documentation . . . . .	78
38.2.1	MB_DualBldNb32 . . . . .	78

<b>39 MB_DualBldNb8.c File Reference</b>	<b>79</b>
39.1 Detailed Description . . . . .	79
39.2 Function Documentation . . . . .	79
39.2.1 MB_DualBldNb8 . . . . .	79
<b>40 MB_DualBldNbb.c File Reference</b>	<b>79</b>
40.1 Detailed Description . . . . .	80
40.2 Function Documentation . . . . .	80
40.2.1 MB_DualBldNbb . . . . .	80
<b>41 MB_Error.c File Reference</b>	<b>80</b>
41.1 Detailed Description . . . . .	80
41.2 Function Documentation . . . . .	81
41.2.1 MB_StrErr . . . . .	81
41.3 Variable Documentation . . . . .	81
41.3.1 err_str . . . . .	81
<b>42 MB_error.h File Reference</b>	<b>81</b>
42.1 Detailed Description . . . . .	81
42.2 Enumeration Type Documentation . . . . .	81
42.2.1 MB_errcode . . . . .	81
42.3 Function Documentation . . . . .	82
42.3.1 MB_StrErr . . . . .	82
<b>43 MB_Frame.c File Reference</b>	<b>82</b>
43.1 Detailed Description . . . . .	82
43.2 Function Documentation . . . . .	82
43.2.1 MB_Frame . . . . .	82
<b>44 MB_HierarBld.c File Reference</b>	<b>83</b>
44.1 Detailed Description . . . . .	83
44.2 Typedef Documentation . . . . .	83
44.2.1 TSWITCHEP . . . . .	83
44.3 Function Documentation . . . . .	83
44.3.1 MB_HierarBld . . . . .	83
<b>45 MB_HierarDualBld.c File Reference</b>	<b>84</b>
45.1 Detailed Description . . . . .	84
45.2 Typedef Documentation . . . . .	84
45.2.1 TSWITCHEP . . . . .	84
45.3 Function Documentation . . . . .	84
45.3.1 MB_HierarDualBld . . . . .	84

<b>46 MB_Histo.c File Reference</b>	<b>85</b>
46.1 Detailed Description . . . . .	85
46.2 Function Documentation . . . . .	85
46.2.1 MB_Histo . . . . .	85
<b>47 MB_Inf.c File Reference</b>	<b>85</b>
47.1 Detailed Description . . . . .	85
47.2 Function Documentation . . . . .	85
47.2.1 MB_Inf . . . . .	85
<b>48 MB_InfFarNb32.c File Reference</b>	<b>86</b>
48.1 Detailed Description . . . . .	86
48.2 Define Documentation . . . . .	86
48.2.1 EDGE_TYPE . . . . .	86
48.3 Function Documentation . . . . .	86
48.3.1 MB_InfFarNb32 . . . . .	86
<b>49 MB_InfFarNb8.c File Reference</b>	<b>87</b>
49.1 Detailed Description . . . . .	87
49.2 Define Documentation . . . . .	87
49.2.1 EDGE_TYPE . . . . .	87
49.3 Function Documentation . . . . .	87
49.3.1 MB_InfFarNb8 . . . . .	87
<b>50 MB_InfFarNbb.c File Reference</b>	<b>88</b>
50.1 Detailed Description . . . . .	88
50.2 Define Documentation . . . . .	88
50.2.1 EDGE_TYPE . . . . .	88
50.3 Function Documentation . . . . .	88
50.3.1 MB_InfFarNbb . . . . .	88
<b>51 MB_InfNb32.c File Reference</b>	<b>89</b>
51.1 Detailed Description . . . . .	89
51.2 Define Documentation . . . . .	89
51.2.1 EDGE_TYPE . . . . .	89
51.3 Function Documentation . . . . .	89
51.3.1 MB_InfNb32 . . . . .	89
<b>52 MB_InfNb8.c File Reference</b>	<b>89</b>
52.1 Detailed Description . . . . .	90
52.2 Define Documentation . . . . .	90
52.2.1 EDGE_TYPE . . . . .	90

52.3	Function Documentation . . . . .	90
52.3.1	MB_InfNb8 . . . . .	90
<b>53</b>	<b>MB_InfNbb.c File Reference</b>	<b>90</b>
53.1	Detailed Description . . . . .	91
53.2	Define Documentation . . . . .	91
53.2.1	EDGE_TYPE . . . . .	91
53.3	Function Documentation . . . . .	91
53.3.1	MB_InfNbb . . . . .	91
<b>54</b>	<b>MB_Inv.c File Reference</b>	<b>91</b>
54.1	Detailed Description . . . . .	92
54.2	Function Documentation . . . . .	92
54.2.1	MB_Inv . . . . .	92
<b>55</b>	<b>MB_Labelb.c File Reference</b>	<b>92</b>
55.1	Detailed Description . . . . .	92
55.2	Define Documentation . . . . .	93
55.2.1	LEFT . . . . .	93
55.2.2	RIGHT . . . . .	93
55.2.3	VAL . . . . .	93
55.3	Typedef Documentation . . . . .	93
55.3.1	TSWITCHEP . . . . .	93
55.4	Function Documentation . . . . .	93
55.4.1	MB_Labelb . . . . .	93
<b>56</b>	<b>MB_LoadExtract.c File Reference</b>	<b>93</b>
56.1	Detailed Description . . . . .	94
56.2	Function Documentation . . . . .	94
56.2.1	MB_Extract . . . . .	94
56.2.2	MB_Extract32 . . . . .	94
56.2.3	MB_Extract8 . . . . .	94
56.2.4	MB_Load . . . . .	95
56.2.5	MB_Load32 . . . . .	95
56.2.6	MB_Load8 . . . . .	95
<b>57</b>	<b>MB_Lookup.c File Reference</b>	<b>95</b>
57.1	Detailed Description . . . . .	95
57.2	Function Documentation . . . . .	96
57.2.1	MB_Lookup . . . . .	96
<b>58</b>	<b>MB_Mask.c File Reference</b>	<b>96</b>

58.1 Detailed Description . . . . .	96
58.2 Function Documentation . . . . .	96
58.2.1 MB_Mask . . . . .	96
<b>59 MB_Mul.c File Reference</b>	<b>96</b>
59.1 Detailed Description . . . . .	97
59.2 Function Documentation . . . . .	97
59.2.1 MB_Mul . . . . .	97
<b>60 MB_Neighbor.c File Reference</b>	<b>97</b>
60.1 Detailed Description . . . . .	97
60.2 Typedef Documentation . . . . .	97
60.2.1 TSWITCHEP . . . . .	97
<b>61 MB_Or.c File Reference</b>	<b>98</b>
61.1 Detailed Description . . . . .	98
61.2 Function Documentation . . . . .	98
61.2.1 MB_Or . . . . .	98
<b>62 MB_Pixel.c File Reference</b>	<b>98</b>
62.1 Detailed Description . . . . .	98
62.2 Function Documentation . . . . .	99
62.2.1 MB_GetPixel . . . . .	99
62.2.2 MB_PutPixel . . . . .	99
<b>63 MB_Range.c File Reference</b>	<b>99</b>
63.1 Detailed Description . . . . .	99
63.2 Function Documentation . . . . .	100
63.2.1 MB_depthRange . . . . .	100
63.2.2 MB_Range . . . . .	100
<b>64 MB_ShftDirection.c File Reference</b>	<b>100</b>
64.1 Detailed Description . . . . .	100
64.2 Typedef Documentation . . . . .	100
64.2.1 TSWITCHEP . . . . .	100
<b>65 MB_Shift32.c File Reference</b>	<b>101</b>
65.1 Detailed Description . . . . .	101
65.2 Define Documentation . . . . .	101
65.2.1 EDGE_TYPE . . . . .	101
65.3 Function Documentation . . . . .	101
65.3.1 MB_Shift32 . . . . .	101

<b>66 MB_Shift8.c File Reference</b>	<b>101</b>
66.1 Detailed Description	102
66.2 Define Documentation	102
66.2.1 EDGE_TYPE	102
66.3 Function Documentation	102
66.3.1 MB_Shift8	102
<b>67 MB_Shiftb.c File Reference</b>	<b>102</b>
67.1 Detailed Description	103
67.2 Define Documentation	103
67.2.1 EDGE_TYPE	103
67.3 Function Documentation	103
67.3.1 MB_Shiftb	103
<b>68 MB_Sub.c File Reference</b>	<b>103</b>
68.1 Detailed Description	104
68.2 Function Documentation	104
68.2.1 MB_Sub	104
<b>69 MB_Sup.c File Reference</b>	<b>104</b>
69.1 Detailed Description	104
69.2 Function Documentation	104
69.2.1 MB_Sup	104
<b>70 MB_SupFarNb32.c File Reference</b>	<b>105</b>
70.1 Detailed Description	105
70.2 Define Documentation	105
70.2.1 EDGE_TYPE	105
70.3 Function Documentation	105
70.3.1 MB_SupFarNb32	105
<b>71 MB_SupFarNb8.c File Reference</b>	<b>106</b>
71.1 Detailed Description	106
71.2 Define Documentation	106
71.2.1 EDGE_TYPE	106
71.3 Function Documentation	106
71.3.1 MB_SupFarNb8	106
<b>72 MB_SupFarNbb.c File Reference</b>	<b>106</b>
72.1 Detailed Description	107
72.2 Define Documentation	107
72.2.1 EDGE_TYPE	107

72.3	Function Documentation . . . . .	107
72.3.1	MB_SupFarNbb . . . . .	107
<b>73</b>	<b>MB_SupMask.c File Reference</b>	<b>107</b>
73.1	Detailed Description . . . . .	108
73.2	Function Documentation . . . . .	108
73.2.1	MB_SupMask . . . . .	108
<b>74</b>	<b>MB_SupNb32.c File Reference</b>	<b>108</b>
74.1	Detailed Description . . . . .	108
74.2	Define Documentation . . . . .	108
74.2.1	EDGE_TYPE . . . . .	108
74.3	Function Documentation . . . . .	109
74.3.1	MB_SupNb32 . . . . .	109
<b>75</b>	<b>MB_SupNb8.c File Reference</b>	<b>109</b>
75.1	Detailed Description . . . . .	109
75.2	Define Documentation . . . . .	109
75.2.1	EDGE_TYPE . . . . .	109
75.3	Function Documentation . . . . .	110
75.3.1	MB_SupNb8 . . . . .	110
<b>76</b>	<b>MB_SupNbb.c File Reference</b>	<b>110</b>
76.1	Detailed Description . . . . .	110
76.2	Define Documentation . . . . .	110
76.2.1	EDGE_TYPE . . . . .	110
76.3	Function Documentation . . . . .	111
76.3.1	MB_SupNbb . . . . .	111
<b>77</b>	<b>MB_Thresh.c File Reference</b>	<b>111</b>
77.1	Detailed Description . . . . .	111
77.2	Function Documentation . . . . .	111
77.2.1	MB_Thresh . . . . .	111
<b>78</b>	<b>MB_Utils.c File Reference</b>	<b>112</b>
78.1	Detailed Description . . . . .	112
78.2	Function Documentation . . . . .	112
78.2.1	MB_aligned_free . . . . .	112
78.2.2	MB_aligned_malloc . . . . .	112
78.2.3	MB_free . . . . .	112
78.2.4	MB_malloc . . . . .	113
78.2.5	MB_memcpy . . . . .	113

78.2.6	MB_memset	113
<b>79</b>	<b>MB_Volume.c File Reference</b>	<b>113</b>
79.1	Detailed Description	114
79.2	Function Documentation	114
79.2.1	MB_Volume	114
79.3	Variable Documentation	114
79.3.1	MB_VolumePerByte	114
<b>80</b>	<b>MB_Watershed.c File Reference</b>	<b>114</b>
80.1	Detailed Description	115
80.2	Typedef Documentation	115
80.2.1	TSWITCHEP	115
80.3	Function Documentation	115
80.3.1	MB_Watershed	115
80.4	Variable Documentation	116
80.4.1	hxNbDir	116
80.4.2	sqNbDir	116
<b>81</b>	<b>MB_Xor.c File Reference</b>	<b>116</b>
81.1	Detailed Description	116
81.2	Function Documentation	116
81.2.1	MB_Xor	116

## List of Figures

1	Image structure and variables . . . . .	117
---	---	-----

Data Structure Index

## 1 Data Structures

Here are the data structures with brief descriptions:

<b>MB_Basins_Ctx</b>	<b>20</b>
<b>MB_Hierarbld_Ctx</b>	<b>22</b>
<b>MB_Hierardualbld_Ctx</b>	<b>23</b>
<b>MB_Image</b>	<b>25</b>
<b>MB_Label</b>	<b>25</b>
<b>MB_ListControl</b>	<b>26</b>
<b>MB_Token</b>	<b>27</b>
<b>MB_Watershed_Ctx</b>	<b>28</b>

File Index

## 2 File List

Here is a list of all documented files with brief descriptions:

<b>mambaApi.h</b>	<b>29</b>
<b>mambaApi_loc.h</b>	<b>52</b>
<b>mambaCommon.h</b>	<b>55</b>
<b>MB_Add.c</b>	<b>58</b>
<b>MB_And.c</b>	<b>59</b>
<b>MB_Basins.c</b>	<b>59</b>
<b>MB_BinHitOrMiss.c</b>	<b>60</b>
<b>MB_BldDirection.c</b>	<b>61</b>
<b>MB_BldNb32.c</b>	<b>61</b>
<b>MB_BldNb8.c</b>	<b>62</b>
<b>MB_BldNbb.c</b>	<b>63</b>
<b>MB_Check.c</b>	<b>64</b>
<b>MB_Compare.c</b>	<b>64</b>
<b>MB_ConAdd.c</b>	<b>65</b>
<b>MB_ConDiv.c</b>	<b>65</b>
<b>MB_ConMul.c</b>	<b>66</b>

MB_ConSet.c	67
MB_ConSub.c	67
MB_Convert.c	68
MB_Copy.c	69
MB_CopyBitPlane.c	70
MB_CopyBytePlane.c	71
MB_Create.c	72
MB_Diff.c	74
MB_DiffNb8.c	74
MB_DiffNbb.c	75
MB_Distanceb.c	76
MB_DualBldNb32.c	78
MB_DualBldNb8.c	79
MB_DualBldNbb.c	79
MB_Error.c	80
MB_error.h	81
MB_Frame.c	82
MB_HierarBld.c	83
MB_HierarDualBld.c	84
MB_Histo.c	85
MB_Inf.c	85
MB_InfFarNb32.c	86
MB_InfFarNb8.c	87
MB_InfFarNbb.c	88
MB_InfNb32.c	89
MB_InfNb8.c	89
MB_InfNbb.c	90
MB_Inv.c	91
MB_Labelb.c	92
MB_LoadExtract.c	93
MB_Lookup.c	95
MB_Mask.c	96

MB_Mul.c	96
MB_Neighbor.c	97
MB_Or.c	98
MB_Pixel.c	98
MB_Range.c	99
MB_ShftDirection.c	100
MB_Shift32.c	101
MB_Shift8.c	101
MB_Shiftb.c	102
MB_Sub.c	103
MB_Sup.c	104
MB_SupFarNb32.c	105
MB_SupFarNb8.c	106
MB_SupFarNbb.c	106
MB_SupMask.c	107
MB_SupNb32.c	108
MB_SupNb8.c	109
MB_SupNbb.c	110
MB_Thresh.c	111
MB_Utils.c	112
MB_Volume.c	113
MB_Watershed.c	114
MB_Xor.c	116

Data Structure Documentation

### 3 MB\_Basins\_Ctx Struct Reference

#### Data Fields

- `UInt32` width
- `UInt32` height
- `MB_Token * TokensArray`
- `MB_ListControl HierarchicalList [256]`
- `PLINE * plines_marker`
- `UInt32 linoff_marker`
- `PLINE * plines_src`
- `UInt32 linoff_src`

- **UInt32** bytes \_marker
- **PIX8** current \_water \_level
- **TSWITCHEP** \* InsertNeighbors

### 3.1 Detailed Description

Structure holding the function contextual information such as the size of the image processed, the pointer to the pixel lines the array of tokens and the current flooding level

### 3.2 Field Documentation

#### 3.2.1 **UInt32** MB \_Basins \_Ctx::bytes \_marker

size in byte of the marker image lines

#### 3.2.2 **PIX8** MB \_Basins \_Ctx::current \_water \_level

Variable indicating which level in the hierarchical list the "water" as attained. Only this level and above can be filled with new tokens.

#### 3.2.3 **UInt32** MB \_Basins \_Ctx::height

The height of the images processed

#### 3.2.4 **MB \_ListControl** MB \_Basins \_Ctx::HierarchicalList[256]

The hierarchical list entries for watershed segmentation

#### 3.2.5 **TSWITCHEP\*** MB \_Basins \_Ctx::InsertNeighbors

meta function which redirects the neighbor function according to the grid

#### 3.2.6 **UInt32** MB \_Basins \_Ctx::linoff \_marker

offset in the marker image lines

#### 3.2.7 **UInt32** MB \_Basins \_Ctx::linoff \_src

offset in the source image lines

#### 3.2.8 **PLINE\*** MB \_Basins \_Ctx::plines \_marker

pointer to the lines of the marker image

#### 3.2.9 **PLINE\*** MB \_Basins \_Ctx::plines \_src

pointer to the lines of the source image

#### 3.2.10 **MB \_Token\*** MB \_Basins \_Ctx::TokensArray

The memory used to hold the elements of the hierarchical list

### 3.2.11 Uint32 MB\_Basins\_Ctx::width

The width of the images processed

The documentation for this struct was generated from the following file:

- MB\_Basins.c

## 4 MB\_Hierarbld\_Ctx Struct Reference

### Data Fields

- Uint32 width
- Uint32 height
- MB\_Token \* TokensArray
- MB\_ListControl HierarchicalList [256]
- Uint32 \* pix\_status
- PLINE \* plines\_mask
- Uint32 linoff\_mask
- PLINE \* plines\_srcdest
- Uint32 linoff\_srcdest
- Uint32 bytes
- PIX8 current\_water\_level
- TSWITCHEP \* InsertNeighbors

### 4.1 Detailed Description

Structure holding the function contextual information such as the size of the processed image, the pointer to the pixel lines, the array of tokens and the current flooding level

### 4.2 Field Documentation

#### 4.2.1 Uint32 MB\_Hierarbld\_Ctx::bytes

size in byte of the image lines

#### 4.2.2 PIX8 MB\_Hierarbld\_Ctx::current\_water\_level

Variable indicating which level in the hierarchical list the "water" as attained. Only this level and above can be filled with new tokens.

#### 4.2.3 Uint32 MB\_Hierarbld\_Ctx::height

The height of the processed images

#### 4.2.4 MB\_ListControl MB\_Hierarbld\_Ctx::HierarchicalList[256]

The hierarchical list entries for watershed segmentation

#### 4.2.5 TSWITCHEP\* MB\_Hierarbld\_Ctx::InsertNeighbors

meta function which redirects the neighbor function according to the grid

#### 4.2.6 `UInt32 MB_Hierarbld_Ctx::linoff_mask`

offset in the mask image lines

#### 4.2.7 `UInt32 MB_Hierarbld_Ctx::linoff_srcdest`

offset in the source/destination image lines

#### 4.2.8 `UInt32* MB_Hierarbld_Ctx::pix_status`

The memory to hold the status of each pixel

#### 4.2.9 `PLINE* MB_Hierarbld_Ctx::plines_mask`

pointer to the lines of the mask image

#### 4.2.10 `PLINE* MB_Hierarbld_Ctx::plines_srcdest`

pointer to the line of the source/destination image

#### 4.2.11 `MB_Token* MB_Hierarbld_Ctx::TokensArray`

The memory used to hold the elements of the hierarchical list

#### 4.2.12 `UInt32 MB_Hierarbld_Ctx::width`

The width of the processed images

The documentation for this struct was generated from the following file:

- `MB_HierarBld.c`

## 5 `MB_Hierardualbld_Ctx` Struct Reference

### Data Fields

- `UInt32 width`
- `UInt32 height`
- `MB_Token * TokensArray`
- `MB_ListControl HierarchicalList [256]`
- `UInt32 * pix_status`
- `PLINE * plines_mask`
- `UInt32 linoff_mask`
- `PLINE * plines_srcdest`
- `UInt32 linoff_srcdest`
- `UInt32 bytes`
- `PIX8 current_water_level`
- `TSWITCHEP * InsertNeighbors`

### 5.1 Detailed Description

Structure holding the function contextual information such as the size of the image processed, the pointer to the pixel lines, the array of tokens and the current flooding level

## 5.2 Field Documentation

### 5.2.1 `UInt32 MB_Hierardualbld_Ctx::bytes`

size in byte of the image lines

### 5.2.2 `PIX8 MB_Hierardualbld_Ctx::current_water_level`

Variable indicating which level in the hierarchical list the "water" as attained. Only this level and above can be filled with new tokens.

### 5.2.3 `UInt32 MB_Hierardualbld_Ctx::height`

The height of the images processed

### 5.2.4 `MB_ListControl MB_Hierardualbld_Ctx::HierarchicalList[256]`

The hierarchical list entries for watershed segmentation

### 5.2.5 `TSWITCHEP* MB_Hierardualbld_Ctx::InsertNeighbors`

meta function which redirects the neighbor function according to the grid

### 5.2.6 `UInt32 MB_Hierardualbld_Ctx::linoff_mask`

offset in the mask image lines

### 5.2.7 `UInt32 MB_Hierardualbld_Ctx::linoff_srcdest`

offset in the source/destination image lines

### 5.2.8 `UInt32* MB_Hierardualbld_Ctx::pix_status`

The memory to hold the status of each pixel

### 5.2.9 `PLINE* MB_Hierardualbld_Ctx::plines_mask`

pointer to the lines of the mask image

### 5.2.10 `PLINE* MB_Hierardualbld_Ctx::plines_srcdest`

pointer to the line of the source/destination image

### 5.2.11 `MB_Token* MB_Hierardualbld_Ctx::TokensArray`

The memory used to hold the elements of the hierarchical list

### 5.2.12 `UInt32 MB_Hierardualbld_Ctx::width`

The width of the images processed

The documentation for this struct was generated from the following file:

- `MB_HierarDualBld.c`

## 6 MB\_Image Struct Reference

```
#include <mambaCommon.h>
```

### Data Fields

- `Uint32 width`
- `Uint32 height`
- `Uint32 depth`
- `PLINE * PLINES`
- `PIX8 * PIXARRAY`

### 6.1 Detailed Description

Images structure with the width, height and depth; the pixels array (`PIXARRAY`) and entry point array to each line of the image (`PLINES`)

### 6.2 Field Documentation

#### 6.2.1 `Uint32 MB_Image::depth`

The depth of the image

#### 6.2.2 `Uint32 MB_Image::height`

The height of the image

#### 6.2.3 `PIX8* MB_Image::PIXARRAY`

pixel array

#### 6.2.4 `PLINE* MB_Image::PLINES`

accessors to pixel lines

#### 6.2.5 `Uint32 MB_Image::width`

The width of the image

The documentation for this struct was generated from the following file:

- `mambaCommon.h`

## 7 MB\_Label Struct Reference

### Data Fields

- `PIX32 * EQ`

- **PIX32 \* CEQ**
- **Uint32 maxEQ**
- **PIX32 current**
- **PIX32 ccurrent**

## 7.1 Detailed Description

Label structure holding all the information needed to handle labels attribution and creation.

## 7.2 Field Documentation

### 7.2.1 PIX32 MB\_Label::ccurrent

Current corrected label index (value given to the next label)

### 7.2.2 PIX32\* MB\_Label::CEQ

Equivalence between corrected label in an image

### 7.2.3 PIX32 MB\_Label::current

Current label index (value given to the next label)

### 7.2.4 PIX32\* MB\_Label::EQ

Equivalence between label in an image

### 7.2.5 Uint32 MB\_Label::maxEQ

The greatest number of label possible according to current image size

The documentation for this struct was generated from the following file:

- **MB\_Labelb.c**

## 8 MB\_ListControl Struct Reference

```
#include <mambaApi_loc.h>
```

### Data Fields

- **int firstx**
- **int firsty**
- **int lastx**
- **int lasty**

## 8.1 Detailed Description

List control structure that gives you the index of the first and last elements of list.

## 8.2 Field Documentation

### 8.2.1 `int MB_ListControl::firstx`

first token of the list (x)

### 8.2.2 `int MB_ListControl::firsty`

first token of the list (y)

### 8.2.3 `int MB_ListControl::lastx`

last token of the list (x)

### 8.2.4 `int MB_ListControl::lasty`

last token of the list (y)

The documentation for this struct was generated from the following file:

- `mambaApi_loc.h`

## 9 MB\_Token Struct Reference

```
#include <mambaApi_loc.h>
```

### Data Fields

- `int nextx`
- `int nexty`

### 9.1 Detailed Description

Token used in hierarchical list. A token gives its next (by position `nextx`, `nexty` in image) token in the list (-1 if the list ends).

### 9.2 Field Documentation

#### 9.2.1 `int MB_Token::nextx`

next token (x)

#### 9.2.2 `int MB_Token::nexty`

next token (y)

The documentation for this struct was generated from the following file:

- `mambaApi_loc.h`

## 10 MB\_Watershed\_Ctx Struct Reference

### Data Fields

- **Uint32** width
- **Uint32** height
- **MB\_Token** \* TokensArray
- **MB\_ListControl** HierarchicalList [256]
- **MB\_ListControl** toreinsertList
- **PLINE** \* plines \_marker
- **Uint32** linoff \_marker
- **PLINE** \* plines \_src
- **Uint32** linoff \_src
- **Uint32** bytes \_marker
- **PIX8** current \_water \_level
- **TSWITCHEP** \* InsertNeighbors

### 10.1 Detailed Description

Structure holding the function contextual information such as the size of the processed image, the pointer to the pixel lines, the array of tokens and the current flooding level

### 10.2 Field Documentation

#### 10.2.1 **Uint32** MB\_Watershed\_Ctx::bytes \_marker

size in byte of the marker image lines

#### 10.2.2 **PIX8** MB\_Watershed\_Ctx::current \_water \_level

Variable indicating which level in the hierarchical list the "water" as attained. Only this level and above can be filled with new tokens.

#### 10.2.3 **Uint32** MB\_Watershed\_Ctx::height

The height of the processed images

#### 10.2.4 **MB\_ListControl** MB\_Watershed\_Ctx::HierarchicalList[256]

The hierarchical list entries for watershed segmentation

#### 10.2.5 **TSWITCHEP\*** MB\_Watershed\_Ctx::InsertNeighbors

meta function which redirects the neighbor function according to the grid

#### 10.2.6 **Uint32** MB\_Watershed\_Ctx::linoff \_marker

offset in the marker image lines

#### 10.2.7 **Uint32** MB\_Watershed\_Ctx::linoff \_src

offset in the source image lines

### 10.2.8 PLINE\* MB\_Watershed\_Ctx::plines\_marker

pointer to the lines of the marker image

### 10.2.9 PLINE\* MB\_Watershed\_Ctx::plines\_src

pointer to the line of the source image

### 10.2.10 MB\_Token\* MB\_Watershed\_Ctx::TokensArray

The memory used to hold the elements of the hierarchical list

### 10.2.11 MB\_ListControl MB\_Watershed\_Ctx::toreinsertList

List of pixels that will be inserted into the hierarchical list if the the parent pixel (their neighbor which is currently processed) is tagged

### 10.2.12 Uint32 MB\_Watershed\_Ctx::width

The width of the processed images

The documentation for this struct was generated from the following file:

- `MB_Watershed.c`

File Documentation

## 11 mambaApi.h File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include <malloc.h>
#include "mambaCommon.h"
#include "MB_error.h"
```

### Functions

- `MB_errcode MB_Create (MB_Image *image, Uint32 width, Uint32 height, Uint32 depth)`
- `MB_errcode MB_Destroy (MB_Image *image)`
- `MB_errcode MB_Load (MB_Image *image, PIX8 *indata, Uint32 len)`
- `MB_errcode MB_Extract (MB_Image *image, PIX8 **outdata, Uint32 *len)`
- `MB_errcode MB_Convert (MB_Image *src, MB_Image *dest)`
- `MB_errcode MB_Copy (MB_Image *src, MB_Image *dest)`
- `MB_errcode MB_CopyLine (MB_Image *src, MB_Image *dest, Uint32 insrc_pos, Uint32 indest_pos)`
- `MB_errcode MB_CropCopy (MB_Image *src, Uint32 x_src, Uint32 y_src, MB_Image *dest, Uint32 x_dest, Uint32 y_dest, Uint32 w, Uint32 h)`
- `MB_errcode MB_PutPixel (MB_Image *dest, Uint32 pixVal, Uint32 x, Uint32 y)`

- **MB\_errcode** **MB\_GetPixel** (**MB\_Image** \*src, **Uint32** \*pixVal, **Uint32** x, **Uint32** y)
- **MB\_errcode** **MB\_And** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Or** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Xor** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Inv** (**MB\_Image** \*src, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Inf** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_InfNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_InfNb8** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_InfNb32** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_InfFarNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_InfFarNb8** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_InfFarNb32** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_Sup** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_SupNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupNb8** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupNb32** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupFarNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupFarNb8** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupFarNb32** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_SupMask** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest, **Uint32** strict)
- **MB\_errcode** **MB\_Add** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Sub** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Mul** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_Shiftb** (**MB\_Image** \*src, **MB\_Image** \*dest, **Uint32** dirnum, **Uint32** count, **Uint32** long\_filler\_pix, enum **MB\_grid\_t** grid)
- **MB\_errcode** **MB\_Shift8** (**MB\_Image** \*src, **MB\_Image** \*dest, **Uint32** dirnum, **Uint32** count, **Uint32** long\_filler\_pix, enum **MB\_grid\_t** grid)
- **MB\_errcode** **MB\_Shift32** (**MB\_Image** \*src, **MB\_Image** \*dest, **Uint32** dirnum, **Uint32** count, **Uint32** long\_filler\_pix, enum **MB\_grid\_t** grid)
- **MB\_errcode** **MB\_Diff** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_DiffNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_DiffNb8** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)
- **MB\_errcode** **MB\_ConAdd** (**MB\_Image** \*src, **Sint32** value, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_ConSub** (**MB\_Image** \*src, **Sint32** value, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_ConMul** (**MB\_Image** \*src, **Uint32** value, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_ConDiv** (**MB\_Image** \*src, **Uint32** value, **MB\_Image** \*dest)
- **MB\_errcode** **MB\_ConSet** (**MB\_Image** \*dest, **Uint32** value)
- **MB\_errcode** **MB\_Volume** (**MB\_Image** \*src, **Uint64** \*pVolume)
- **MB\_errcode** **MB\_Check** (**MB\_Image** \*src, **Uint32** \*isEmpty)
- **MB\_errcode** **MB\_Lookup** (**MB\_Image** \*src, **MB\_Image** \*dest, **Uint32** \*ptab)
- **MB\_errcode** **MB\_Histo** (**MB\_Image** \*src, **Uint32** \*phisto)

- **MB\_errcode MB\_Compare** (**MB\_Image \*src**, **MB\_Image \*cmp**, **MB\_Image \*dest**, **Sint32 \*px**, **Sint32 \*py**)
- **MB\_errcode MB\_Thresh** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 low**, **Uint32 high**)
- **MB\_errcode MB\_BldNbb** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_BldNb8** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_BldNb32** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_HierarBld** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_DualBldNbb** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_DualBldNb8** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_DualBldNb32** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **Uint32 dirnum**, **Uint64 \*pVolume**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_HierarDualBld** (**MB\_Image \*mask**, **MB\_Image \*srcdest**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_Mask** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 maskf**, **Uint32 maskt**)
- **MB\_errcode MB\_Range** (**MB\_Image \*src**, **Uint32 \*min**, **Uint32 \*max**)
- **MB\_errcode MB\_depthRange** (**MB\_Image \*src**, **Uint32 \*min**, **Uint32 \*max**)
- **MB\_errcode MB\_CopyBitPlane** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 plane**)
- **MB\_errcode MB\_CopyBytePlane** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 plane**)
- **MB\_errcode MB\_BinHitOrMiss** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 es0**, **Uint32 es1**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_Labelb** (**MB\_Image \*src**, **MB\_Image \*dest**, **Uint32 lblow**, **Uint32 lbhigh**, **Uint32 \*pNbobj**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_Distanceb** (**MB\_Image \*src**, **MB\_Image \*dest**, **enum MB\_grid\_t grid**, **enum MB\_edgemode\_t edge**)
- **MB\_errcode MB\_Watershed** (**MB\_Image \*src**, **MB\_Image \*marker**, **Uint32 max\_level**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_Basins** (**MB\_Image \*src**, **MB\_Image \*marker**, **Uint32 max\_level**, **enum MB\_grid\_t grid**)
- **MB\_errcode MB\_Frame** (**MB\_Image \*src**, **Uint32 thresval**, **Uint32 \*ulx**, **Uint32 \*uly**, **Uint32 \*brx**, **Uint32 \*bry**)

## Variables

- **Uint32 MB\_refcounter**

## 11.1 Detailed Description

### Date

11-4-2007

This file contains the various definitions, global variables macro, struct and functions created for the library.

## 11.2 Function Documentation

### 11.2.1 **MB\_errcode MB\_Add** (**MB\_Image \* src1**, **MB\_Image \* src2**, **MB\_Image \* dest**)

Adds the pixels of two images and put the result in the third image. Depending on the format of the target image, the result may be saturated or not. You can perform the following additions : 1-bit + 1-bit = 1-bit (call the MB\_Or function) 1-bit + 8-bit = 8-bit (saturated) 1-bit + 8-bit = 32-bit 1-bit + 32-bit = 32-bit 8-bit + 8-bit = 8-bit (saturated) 8-bit + 8-bit = 32-bit 8-bit + 32-bit = 32-bit 32-bit + 32-bit = 32-bit

### Parameters

*src1* image 1  
*src2* image 2  
*dest* image resulting of the addition of image 1 and 2.

### Returns

An error code (NO\_ERR if successful)

**11.2.2** `MB_errcode MB_And ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Performs a bitwise AND between the pixels of two images.

### Parameters

*src1* image 1  
*src2* image 2  
*dest* destination image

### Returns

An error code (NO\_ERR if successful)

**11.2.3** `MB_errcode MB_Basins ( MB_Image * src, MB_Image * marker, Uint32 max_level, enum MB_grid_t grid )`

Performs a watershed segmentation of the image using the marker image as a starting point for the flooding. The function returns the catchment basins of the watershed but no actual watershed line. It is recommended to use this functions rather than MB\_Watershed if you are only interested in catchment basins (faster). The result is put into a the 32-bits marker image.

The segmentation is coded as follows into the 32 bit values. | 0 | 1 | 2 | 3 | |-----|-----|-----|-----| | Segment label | Unused | Each byte can be accessed using the function MB\_CopyBytePlane.

### Parameters

*src* the greyscale image to be segmented  
*marker* the marker image in which the result of segmentation will be put  
*max\_level* the maximum level reached by the water.  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

**11.2.4** `MB_errcode MB_BinHitOrMiss ( MB_Image * src, MB_Image * dest, Uint32 es0, Uint32 es1, enum MB_grid_t grid )`

Performs a binary Hit-or-Miss operation on image imIn using the structuring elements es0 and es1. Structuring elements are integer values coding which direction must be taken into account. es0 indicating which neighbor of the current pixel will be checked for 0 value. es1 those which will be evaluated for 1 value.

For example, in hexagonal grid, it means that if you want to look for a pattern where the neighbors in direction 6 and 1 are true while the current pixel is false just as neighbors 2 and 5, you will encode this in the elements es0 and es1 like this : 1 1 0 0 0 X X es0 = 1+4+32 es1 = 64+2

### Parameters

*src* output image  
*dest* input image (must be different of *src*)  
*es0* structuring element for 0 value.  
*es1* structuring element for 1 value.  
*grid* grid configuration

### Returns

An error code (NO\_ERR if successful)

**11.2.5** `MB_errcode MB_BldNb32 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

**11.2.6** `MB_errcode MB_BldNb8 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

**11.2.7** `MB_errcode MB_BldNb6 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

### Parameters

*mask* the mask image

*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.8 MB\_errcode MB\_Check ( MB\_Image \* *src*, Uint32 \* *isEmpty* )

Verifies that the image is not empty (all pixels to 0).

#### Parameters

*src* the source image  
*isEmpty* an integer which is set to 1 if empty or 0 if not

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.9 MB\_errcode MB\_Compare ( MB\_Image \* *src*, MB\_Image \* *cmp*, MB\_Image \* *dest*, Sint32 \* *px*, Sint32 \* *py* )

Performs a comparison between a source image and a given base image.

#### Parameters

*src* the source image  
*cmp* the image to which the source image is compared  
*dest* destination image  
*px* position in x of the first different pixel between the two images (-1 if images are similar)  
*py* position in y of the first different pixel between the two images (-1 if images are similar)

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.10 MB\_errcode MB\_ConAdd ( MB\_Image \* *src*, Sint32 *value*, MB\_Image \* *dest* )

Adds a constant value to the pixels of an image.

#### Parameters

*src* the source image  
*value* the constant value to be added to the pixels  
*dest* the image resulting of the addition of image 1 and value.

#### Returns

An error code (NO\_ERR if successful)

**11.2.11** `MB_errcode MB_ConDiv ( MB_Image * src, Uint32 value, MB_Image * dest )`

Divides (quotient) the pixels of an image by a constant value.

**Parameters**

*src* the source image

*value* the constant value used in the division

*dest* the image resulting of the division of image 1 by the value.

**Returns**

An error code (NO\_ERR if successful)

**11.2.12** `MB_errcode MB_ConMul ( MB_Image * src, Uint32 value, MB_Image * dest )`

Multiplies a constant value to the pixels of an image.

**Parameters**

*src* the source image

*value* the constant value to be multiplied to the pixels

*dest* the image resulting of the multiplication of image 1 by the value.

**Returns**

An error code (NO\_ERR if successful)

**11.2.13** `MB_errcode MB_ConSet ( MB_Image * dest, Uint32 value )`

Fills an image with a specific value

**Parameters**

*dest* the image

*value* the value to fill the image

**Returns**

An error code (NO\_ERR if successful)

**11.2.14** `MB_errcode MB_ConSub ( MB_Image * src, Sint32 value, MB_Image * dest )`

Subtracts a constant value to the pixels of an image.

**Parameters**

*src* the source image

*value* the constant value to be subtracted to the pixels

*dest* the image resulting of the subtraction of image 1 and value.

**Returns**

An error code (NO\_ERR if successful)

### 11.2.15 MB\_errcode MB\_Convert ( MB\_Image \* *src*, MB\_Image \* *dest* )

Converts an image of a given depth into another depth

This function does not depend on the window computation currently set.

#### Parameters

*src* source image

*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

### 11.2.16 MB\_errcode MB\_Copy ( MB\_Image \* *src*, MB\_Image \* *dest* )

Copies an image data contents into another image This copy works with same size images.

#### Parameters

*src* the source image

*dest* the destination image

#### Returns

An error code (NO\_ERR if successful)

### 11.2.17 MB\_errcode MB\_CopyBitPlane ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 *plane* )

Inserts or extracts the bit plane in/out an image *src* into *dest*.

#### Parameters

*src* source image

*dest* destination image

*plane* the plane number

#### Returns

An error code (NO\_ERR if successful)

### 11.2.18 MB\_errcode MB\_CopyBytePlane ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 *plane* )

Inserts or extracts the byte plane in/out an image *src* into/from *dest*.

#### Parameters

*src* source image

*dest* destination image

*plane* the plane number

#### Returns

An error code (NO\_ERR if successful)

**11.2.19** `MB_errcode MB_CopyLine ( MB_Image * src, MB_Image * dest, Uint32 insrc_pos, Uint32 indest_pos )`

Copies an image line contents into another image line

**Parameters**

- src* the source image
- dest* the destination image
- insrc\_pos* the position of the line copied from src
- indest\_pos* the position in dest in which the line is copied

**Returns**

An error code (NO\_ERR if successful)

**11.2.20** `MB_errcode MB_Create ( MB_Image * image, Uint32 width, Uint32 height, Uint32 depth )`

Creates an image (memory allocation) with the correct size and depth given as argument. The size is deduced from the requested size given in argument. The size must be a multiple of MB\_ROUND\_W for width and MB\_ROUND\_H for height. The size cannot be greater than MB\_MAX\_IMAGE\_W\_SIZExMB\_MAX\_IMAGE\_H\_SIZE.

**Parameters**

- image* the created image
- width* the width of the created image
- height* the height of the created image
- depth* the depth of the created image

**Returns**

An error code (NO\_ERR if successful)

**11.2.21** `MB_errcode MB_CropCopy ( MB_Image * src, Uint32 x_src, Uint32 y_src, MB_Image * dest, Uint32 x_dest, Uint32 y_dest, Uint32 w, Uint32 h )`

Copies an image data contents into another image. This copy can work with image of different sizes. As the size can be different the position where the copy occurs must be specified for both images as well as the size of the copy. The function will compute the actual crop inside the source and destination images. Works only with non binary images.

**Parameters**

- src* the source image
- x\_src* the x position in the source image where the copy should begin
- y\_src* the y position in the source image where the copy should begin
- dest* the destination image
- x\_dest* the x position in the destination image where the copy should happen
- y\_dest* the y position in the destination image where the copy should happen
- w* the width of the copy
- h* the height of the copy

**Returns**

An error code (NO\_ERR if successful)

### 11.2.22 `MB_errcode MB_depthRange ( MB_Image * src, Uint32 * min, Uint32 * max )`

gives the minimum and maximum possible values of the image pixels given the image depth

#### Parameters

- src* source image
- min* the minimum possible value of the pixels
- max* the maximum possible value of the pixels

#### Returns

An error code (NO\_ERR if successful)

### 11.2.23 `MB_errcode MB_Destroy ( MB_Image * image )`

Destroys an image (memory freeing)

#### Parameters

- image* the image to be destroyed

#### Returns

An error code (NO\_ERR if successful)

### 11.2.24 `MB_errcode MB_Diff ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Computes the set difference between two images. The result image pixel value is the pixel value of image 1 if this value was greater than value of pixel 2 otherwise the minimum possible value is set for the pixel

#### Parameters

- src1* source image 1
- src2* source image 2
- dest* destination image

#### Returns

An error code (NO\_ERR if successful)

### 11.2.25 `MB_errcode MB_DiffNb8 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Computes the set difference between two greyscale image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

- src* source image in which the neighbors are taken
- srcdest* source of the central pixel and destination image
- nbrnum* the neighbor index
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.26** `MB_errcode MB_DiffNbb ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Computes the set difference between two binary image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbors are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.27** `MB_errcode MB_Distanceb ( MB_Image * src, MB_Image * dest, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Computes for each pixel the distance to the edge of the set in which the pixel is found

The algorithm works with two passes, first from the top left to the bottom right and then back.

#### Parameters

*src* the binary source image  
*dest* the 32-bits image in which the distance for each pixel is stored  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.28** `MB_errcode MB_DualBldNb32 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds (dual operation) an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

**11.2.29** `MB_errcode MB_DualBldNb8 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds (dual operation) an image according to a direction and a mask image. The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

#### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (`NO_ERR` if successful)

**11.2.30** `MB_errcode MB_DualBldNbb ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

#### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (`NO_ERR` if successful)

**11.2.31** `MB_errcode MB_Extract ( MB_Image * image, PIX8 ** outdata, Uint32 * len )`

Reads an image data contents and put it in an array

#### Parameters

*image* the image to read  
*outdata* pointer to the array created (malloc) and filled with the pixel data of the image  
*len* the length in bytes of data extracted

#### Returns

An error code (`NO_ERR` if successful)

**11.2.32** `MB_errcode MB_Frame ( MB_Image * src, Uint32 thresval, Uint32 * ulx, Uint32 * uly, Uint32 * brx, Uint32 * bry )`

Returns the smallest frame that contains all the pixels of image that are greater or equal to the given threshold value, using the four last pointers to describe it

### Parameters

*src* source image  
*thresval* the threshold value used to compute the frame  
*ulx* the x-coordinate of the upper left corner of the frame  
*uly* the y-coordinate of the upper left corner of the frame  
*brx* the x-coordinate of the bottom right corner of the frame  
*bry* the y-coordinate of the bottom right corner of the frame

### Returns

An error code (NO\_ERR if successful)

**11.2.33** MB\_errcode MB\_GetPixel ( MB\_Image \* *src*, Uint32 \* *pixVal*, Uint32 *x*, Uint32 *y* )

Gets the pixel value inside the image at the given position

### Parameters

*src* the image  
*pixVal* the returned pixel value  
*x* position in x of the pixel targeted  
*y* position in y of the pixel targeted

### Returns

An error code (NO\_ERR if successful)

**11.2.34** MB\_errcode MB\_HierarBld ( MB\_Image \* *mask*, MB\_Image \* *srcdest*, enum MB\_grid\_t *grid* )

(re)Builds an image according to a mask image and using a hierarchical list to compute the rebuild.

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

**11.2.35** MB\_errcode MB\_HierarDualBld ( MB\_Image \* *mask*, MB\_Image \* *srcdest*, enum MB\_grid\_t *grid* )

(re)Builds (dual operation) an image according to a mask image and using a hierarchical list to compute the rebuild.

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

### 11.2.36 MB\_errcode MB\_Histo ( MB\_Image \* *src*, Uint32 \* *phisto* )

Computes the histogram of an image. The histogram is an array with a minimal size of 256.

#### Parameters

*src* source image  
*phisto* pointer to the histogram array

#### Returns

An error code (NO\_ERR if successful)

### 11.2.37 MB\_errcode MB\_Inf ( MB\_Image \* *src1*, MB\_Image \* *src2*, MB\_Image \* *dest* )

Determines the inferior value between the pixels of two images. The result is put in the corresponding pixel position in the destination image.

#### Parameters

*src1* image 1  
*src2* image 2  
*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

### 11.2.38 MB\_errcode MB\_InfFarNb32 ( MB\_Image \* *src*, MB\_Image \* *srcdest*, Uint32 *nbrnum*, Uint32 *count*, enum MB\_grid\_t *grid*, enum MB\_edgemode\_t *edge* )

Looks for the minimum between two 32-bits image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

### 11.2.39 MB\_errcode MB\_InfFarNb8 ( MB\_Image \* *src*, MB\_Image \* *srcdest*, Uint32 *nbrnum*, Uint32 *count*, enum MB\_grid\_t *grid*, enum MB\_edgemode\_t *edge* )

Looks for the minimum between two greyscale image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken

*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.40** `MB_errcode MB_InfFarNbb ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two binary image pixels (a central pixel and its far neighbor in the other image)  
 The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.41** `MB_errcode MB_InfNb32 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two 32-bits image pixels (a central pixel and its neighbor in the other image)  
 The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srcdest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.42** `MB_errcode MB_InfNb8 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two greyscale image pixels (a central pixel and its neighbor in the other image)  
 The neighbor depends on the grid used (see MB\_ngh.h).

### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srcdest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixel near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

**11.2.43** `MB_errcode MB_InfNbb ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two binary image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srcdest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

**11.2.44** `MB_errcode MB_Inv ( MB_Image * src, MB_Image * dest )`

Inverts the pixels values (negation) of the source image.

### Parameters

*src* source image  
*dest* destination image

### Returns

An error code (NO\_ERR if successful)

**11.2.45** `MB_errcode MB_Labelb ( MB_Image * src, MB_Image * dest, Uint32 lblow, Uint32 lbhigh, Uint32 * pNboj, enum MB_grid_t grid )`

Labeling the object found in src image.

### Parameters

*src* the binary source image where the object must be labelled  
*dest* the 32-bit image where object are labelled  
*lblow* the lowest value allowed for label on the low byte (must be inferior to lbhigh)

*lbhigh* the first high value NOT allowed for label on the low byte (maximum allowed is 256)

*pNobj* the number of object founds

*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.46 MB\_errcode MB\_Load ( MB\_Image \* *image*, PIX8 \* *indata*, Uint32 *len* )

Loads an image data with data given in argument

#### Parameters

*image* the image to fill

*indata* the data to fill the image with (complete pixels values)

*len* the length of data given

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.47 MB\_errcode MB\_Lookup ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 \* *ptab* )

Applies the function in the lookup table to the pixels of source image. A pixel value is changed to a new value accordingly with its current value

#### Parameters

*src* source image

*dest* destination image

*ptab* the lookup table pointer

#### Returns

An error code (NO\_ERR if successful)

#### 11.2.48 MB\_errcode MB\_Mask ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 *maskf*, Uint32 *maskt* )

Converts a binary image in a grey scale image (8-bits) or in a 32-bits image using value *maskf* to replace 0 and *maskt* to replace 1.

#### Parameters

*src* binary source image

*dest* destination image

*maskf* for 0 (false) pixel value

*maskt* for 1 (true) pixel value

#### Returns

An error code (NO\_ERR if successful)

**11.2.49** `MB_errcode MB_Mul ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Multiplies the pixels of two images and puts the result in the third image. Depending on the format of the target image, the result may be saturated or not. You can perform the following additions : 1-bit \* 1-bit = 1-bit (call the `MB_And` function) 1-bit \* 8-bit = 8-bit 1-bit \* 8-bit = 32-bit 1-bit \* 32-bit = 32-bit 8-bit \* 8-bit = 8-bit (saturated) 8-bit \* 8-bit = 32-bit 8-bit \* 32-bit = 32-bit 32-bit \* 32-bit = 32-bit

#### Parameters

*src1* image 1

*src2* image 2

*dest* image resulting of the multiplication of image 1 and 2.

#### Returns

An error code (`NO_ERR` if successful)

**11.2.50** `MB_errcode MB_Or ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Applies a bitwise OR on the pixels of two images. All the images must have the same depth for correct work.

#### Parameters

*src1* image 1

*src2* image 2

*dest* destination image

#### Returns

An error code (`NO_ERR` if successful)

**11.2.51** `MB_errcode MB_PutPixel ( MB_Image * dest, Uint32 pixVal, Uint32 x, Uint32 y )`

Puts the pixel value inside the image at the given position

#### Parameters

*dest* the image

*pixVal* the pixel value

*x* position in x of the pixel targeted

*y* position in y of the pixel targeted

#### Returns

An error code (`NO_ERR` if successful)

**11.2.52** `MB_errcode MB_Range ( MB_Image * src, Uint32 * min, Uint32 * max )`

gives the minimum and maximum values of the image pixels i.e its range.

#### Parameters

*src* source image

*min* the minimum value of the pixels

*max* the maximum value of the pixels

#### Returns

An error code (NO\_ERR if successful)

**11.2.53** `MB_errcode MB_Shift32 ( MB_Image * src, MB_Image * dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid )`

Shifts the content of a 32-bits image in a given direction with a given amplitude The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*src* source image

*dest* destination image

*dirnum* the direction index

*count* the amplitude of the shift

*long\_filler\_pix* the value used to fill the created space

*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

**11.2.54** `MB_errcode MB_Shift8 ( MB_Image * src, MB_Image * dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid )`

Shifts the contents of a 8-bits image in a given direction with a given amplitude The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*src* source image

*dest* destination image

*dirnum* the direction index

*count* the amplitude of the shift

*long\_filler\_pix* the value used to fill the created space

*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

**11.2.55** `MB_errcode MB_Shiftb ( MB_Image * src, MB_Image * dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid )`

Shifts the source image in the given direction. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*src* source image

*dest* destination image

*dirnum* the direction(from 0 to 8)

*count* the amplitude of the shift (in pixels)  
*long\_filler\_pix* the value used to fill the created space  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

**11.2.56** `MB_errcode MB_Sub ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Subtracts the values of pixels of the second image to the values of the pixels in the first image

#### Parameters

*src1* image 1  
*src2* image 2  
*dest* image resulting of the subtraction

#### Returns

An error code (NO\_ERR if successful)

**11.2.57** `MB_errcode MB_Sup ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Determines the superior value between the pixels of two images. The result is put in the corresponding pixel position in the destination image.

#### Parameters

*src1* image 1  
*src2* image 2  
*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

**11.2.58** `MB_errcode MB_SupFarNb32 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two 32-bits image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.59** `MB_errcode MB_SupFarNb8 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two grey scale image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.60** `MB_errcode MB_SupFarNbb ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two binary image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.61** `MB_errcode MB_SupMask ( MB_Image * src1, MB_Image * src2, MB_Image * dest, Uint32 strict )`

Computes a binary image where pixels are set to 1 when the pixels of image 1 have greater values than pixels of image 2 otherwise 0

#### Parameters

*src1* source image 1  
*src2* source image 2  
*dest* destination image  
*strict* flag indicating if the comparison is strict or large

#### Returns

An error code (NO\_ERR if successful)

**11.2.62** `MB_errcode MB_SupNb32 ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two 32-bits image pixels (a central pixel and its neighbor in the other image)  
The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srctest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srctest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.63** `MB_errcode MB_SupNb8 ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two greys cale image pixels (a central pixel and its neighbor in the other image)  
The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srctest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srctest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.64** `MB_errcode MB_SupNbb ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two binary image pixels (a central pixel and its neighbor in the other image)  
The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srctest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if src and srctest are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

**11.2.65** `MB_errcode MB_Thresh ( MB_Image * src, MB_Image * dest, Uint32 low, Uint32 high )`

Fills a binary image according to the following rules : if pixel value lower than *low* or higher than *high* the binary pixel is set to 0, in other cases the pixel is set to 1.

**Parameters**

*src* source image  
*dest* destination image  
*low* low value for threshold  
*high* high value for treshold

**Returns**

An error code (NO\_ERR if successful)

**11.2.66** `MB_errcode MB_Volume ( MB_Image * src, Uint64 * pVolume )`

Computes the volume of an image. The volume is the sum of the pixel values (i.e. integration of the image)

**Parameters**

*src* source image  
*pVolume* pointer to the volume variable

**Returns**

An error code (NO\_ERR if successful)

**11.2.67** `MB_errcode MB_Watershed ( MB_Image * src, MB_Image * marker, Uint32 max_level, enum MB_grid_t grid )`

Performs a watershed segmentation of the image using the marker image as a starting point for the flooding. The function builds the actual watershed line (idempotent) plus catchment basins (not idempotent). The result is put into the 32-bits marker image.

The segmentation is coded as follows into the 32-bits values. | 0 | 1 | 2 | 3 | |-----|-----|-----|-----| | Segment label | isLine | Each byte can be accessed using the function MB\_CopyBytePlane. isLine is a value indicating if the pixel belongs to the watershed (255 if this is the case, undefined otherwise).

**Parameters**

*src* the greyscale image to segment  
*marker* the marker image in which the result of segmentation will be put  
*max\_level* the maximum level reach by the water.  
*grid* the grid used (either square or hexagonal)

**Returns**

An error code (NO\_ERR if successful)

**11.2.68** `MB_errcode MB_Xor ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Applies a XOR on the pixels of two images. All the images must have the same depth for correct work.

## Parameters

- src1* image 1
- src2* image 2
- dest* destination image

## Returns

An error code (NO\_ERR if successful)

## 11.3 Variable Documentation

### 11.3.1 Uint32 MB\_refcounter

image counter

## 12 mambaApi\_loc.h File Reference

```
#include "mambaApi.h"
```

## Data Structures

- struct MB\_Token
- struct MB\_ListControl

## Defines

- #define MB\_LIST\_END -1
- #define MB\_PROBE\_PAIR(im\_in, im\_out) (((im\_in->depth)<<7) + (im\_out->depth))
- #define MB\_CHECK\_SIZE\_2(im1, im2) (((im1->width)==(im2->width))&&((im1->height)==(im2->height)))
- #define MB\_CHECK\_SIZE\_3(im1, im2, im3) (MB\_CHECK\_SIZE\_2(im1, im2) && MB\_CHECK\_SIZE\_2(im1, im3))

## Functions

- void \* MB\_malloc (int size)
- void \* MB\_aligned\_malloc (int size, int alignment)
- void MB\_free (void \*ptr)
- void MB\_aligned\_free (void \*ptr)
- void \* MB\_memset (void \*s, int c, int size)
- void \* MB\_memcpy (void \*dest, const void \*src, int size)

## Variables

- const int sqNbDir [9][2]
- const int hxNbDir [2][7][2]
- const Uint64 MB\_VolumePerByte [256]

## 12.1 Detailed Description

### Date

11-4-2007

This file contains the various definitions, global variables macro, struct and functions that are shared between components of the library but are not meant to be exported to the outside world.

## 12.2 Define Documentation

**12.2.1** `#define MB_CHECK_SIZE_2( im1, im2 ) (((im1->width)==(im2->width))&&((im1->height)==(im2->height)))`

Returns True if the two images sizes are compatibles

**12.2.2** `#define MB_CHECK_SIZE_3( im1, im2, im3 ) (MB_CHECK_SIZE_2(im1, im2) && MB_CHECK_SIZE_2(im1, im3))`

Returns True if the three images sizes are compatibles

**12.2.3** `#define MB_LIST_END -1`

Value used to specify the end of a hierarchical list

**12.2.4** `#define MB_PROBE_PAIR( im_in, im_out ) (((im_in->depth)<<7) + (im_out->depth))`

Returns the value of the images combination MB\_PAIR\_x\_x

## 12.3 Function Documentation

**12.3.1** `void MB_aligned_free ( void * ptr )`

Frees aligned memory.

### Parameters

*ptr* pointer to the memory space to free

**12.3.2** `void* MB_aligned_malloc ( int size, int alignment )`

Allocates aligned memory (faster and safer access for SSE2 instructions for example).

### Parameters

*size* size in bytes of the allocated memory

*alignment* value to which the memory space is aligned

### Returns

a pointer to the memory space or NULL if unsuccessful

### 12.3.3 void MB\_free ( void \* *ptr* )

Frees memory.

#### Parameters

*ptr* pointer to the memory space to free

### 12.3.4 void\* MB\_malloc ( int *size* )

Allocates memory.

#### Parameters

*size* size in byte of the allocated memory

#### Returns

a pointer to the memory space or NULL if unsuccessful

### 12.3.5 void\* MB\_memcpy ( void \* *dest*, const void \* *src*, int *size* )

Copies a memory space to another. This copy should be protected for safe aliased memory copy.

#### Parameters

*dest* pointer to the destination memory space

*src* pointer to the source memory space

*size* the size of the copy

#### Returns

a pointer to the destination memory space

### 12.3.6 void\* MB\_memset ( void \* *s*, int *c*, int *size* )

Sets a memory space to a specific value.

#### Parameters

*s* pointer to the memory space to set

*c* value to use to set

*size* the memory space size

#### Returns

a pointer to the memory space set

## 12.4 Variable Documentation

### 12.4.1 const int hxNbDir[2][7][2]

Table giving the offset for the neighbor in hexagonal grid (x and y)

### 12.4.2 const Uint64 MB\_VolumePerByte[256]

Volume arrays

### 12.4.3 `const int sqNbDir[9][2]`

Table giving the offset for the neighbor in square grid (x and y)

## 13 mambaCommon.h File Reference

```
#include <stdint.h>
```

### Data Structures

- struct `MB_Image`

### Defines

- `#define MB_X_LEFT(im) X_LEFT`
- `#define MB_X_RIGHT(im) X_RIGHT`
- `#define MB_Y_TOP(im) Y_TOP`
- `#define MB_Y_BOTTOM(im) Y_BOTTOM`
- `#define MB_LINE_COUNT(im) ((im->width*im->depth)/CHARBIT)`
- `#define MB_LINE_OFFSET(im) (MB_X_LEFT(im))`
- `#define BIN_FILL_VALUE(edge) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`
- `#define GREY_FILL_VALUE(edge) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`
- `#define I32_FILL_VALUE(edge) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`

### Typedefs

- `typedef uint8_t UInt8`
- `typedef uint16_t UInt16`
- `typedef uint32_t UInt32`
- `typedef uint64_t UInt64`
- `typedef int8_t Sint8`
- `typedef int16_t Sint16`
- `typedef int32_t Sint32`
- `typedef int64_t Sint64`
- `typedef uint8_t PIX8`
- `typedef PIX8 * PLINE`
- `typedef uint32_t PIX32`
- `typedef PIX32 * PLINE32`

### Enumerations

- `enum MB_grid_t { MB_HEXAGONAL_GRID = 1, MB_SQUARE_GRID = 0 }`
- `enum MB_edgemode_t { MB_EMPTY_EDGE = 0, MB_FILLED_EDGE = 1 }`

### 13.1 Detailed Description

#### Date

31-03-2009

This file contains the various definitions, macro, struct that are commons between the various modules of the library

The copyright license of Mamba is reminded here :

Copyright (c) <2009>, <Nicolas BEUCHER and ARMINES for the Centre de Morphologie Mathématique(CMM), common research center to ARMINES and MINES Paristech>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

Except as contained in this notice, the names of the above copyright holders shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without their prior written authorization.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 13.2 Define Documentation

**13.2.1** `#define BIN_FILL_VALUE( edge ) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`

How to fill the edge (binary bits images)

**13.2.2** `#define GREY_FILL_VALUE( edge ) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`

How to fill the edge (8 bits images)

**13.2.3** `#define I32_FILL_VALUE( edge ) ((edge==MB_FILLED_EDGE) ? UINT32_MAX:0)`

How to fill the edge (32 bits images)

**13.2.4** `#define MB_LINE_COUNT( im ) ((im->width*im->depth)/CHARBIT)`

Returns the size in bytes of an image line

**13.2.5** `#define MB_LINE_OFFSET( im ) (MB_X_LEFT(im))`

Returns the size in bytes of the line offset

**13.2.6** `#define MB_X_LEFT( im ) X_LEFT`

Getting image frame offset from left

**13.2.7** `#define MB_X_RIGHT( im ) X_RIGHT`

Getting image frame offset from right

**13.2.8 #define MB\_Y\_BOTTOM( *im* ) Y\_BOTTOM**

Getting image frame offset from bottom

**13.2.9 #define MB\_Y\_TOP( *im* ) Y\_TOP**

Getting image frame offset from top

**13.3 Typedef Documentation****13.3.1 typedef uint32\_t PIX32**

Signed 32-bit pixels value type

**13.3.2 typedef uint8\_t PIX8**

grey-scale pixels value type

**13.3.3 typedef PIX8\* PLINE**

Pixels line pointers type

**13.3.4 typedef PIX32\* PLINE32**

32-bit pixels line pointers type

**13.3.5 typedef int16\_t Sint16**

Signed 16 bit value type

**13.3.6 typedef int32\_t Sint32**

Signed 32 bit value type

**13.3.7 typedef int64\_t Sint64**

Signed 64 bit value type

**13.3.8 typedef int8\_t Sint8**

Signed 8 bit value type

**13.3.9 typedef uint16\_t Uint16**

Unsigned 16 bit value type

**13.3.10 typedef uint32\_t Uint32**

Unsigned 32 bit value type

### 13.3.11 typedef uint64\_t Uint64

Unsigned 64 bit value type

### 13.3.12 typedef uint8\_t Uint8

Unsigned 8 bit value type

## 13.4 Enumeration Type Documentation

### 13.4.1 enum MB\_\_edgemode\_t

enumerate for edge mode : either empty or filled

### 13.4.2 enum MB\_\_grid\_t

enumerate for grid : either square or hexagonal the value is supposed to give the number of directions allowed

## 14 MB\_\_Add.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB__errcode MB__Add (MB__Image *src1, MB__Image *src2, MB__Image *dest)`

### 14.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-2-2007

### 14.2 Function Documentation

#### 14.2.1 `MB__errcode MB__Add ( MB__Image * src1, MB__Image * src2, MB__Image * dest )`

Adds the pixels of two images and put the result in the third image. Depending on the format of the target image, the result may be saturated or not. You can perform the following additions : 1-bit + 1-bit = 1-bit (call the MB\_\_Or function) 1-bit + 8-bit = 8-bit (saturated) 1-bit + 8-bit = 32-bit 1-bit + 32-bit = 32-bit 8-bit + 8-bit = 8-bit (saturated) 8-bit + 8-bit = 32-bit 8-bit + 32-bit = 32-bit 32-bit + 32-bit = 32-bit

#### Parameters

*src1* image 1

*src2* image 2

*dest* image resulting of the addition of image 1 and 2.

#### Returns

An error code (NO\_\_ERR if successful)

## 15 MB\_And.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_And (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

### 15.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-2-2007

### 15.2 Function Documentation

**15.2.1** `MB_errcode MB_And ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Performs a bitwise AND between the pixels of two images.

#### Parameters

- *src1* image 1
- *src2* image 2
- *dest* destination image

#### Returns

An error code (NO\_ERR if successful)

## 16 MB\_Basins.c File Reference

```
#include "mambaApi_loc.h"
```

### Data Structures

- struct `MB_Basins_Ctx`

### Typedefs

- typedef void( `TSWITCHEP` )(void \*ctx, int x, int y)

### Functions

- `MB_errcode MB_Basins (MB_Image *src, MB_Image *marker, Uint32 max_level, enum MB_grid_t grid)`

## 16.1 Detailed Description

### Author

Nicolas Beucher

### Date

01-02-2009

## 16.2 Typedef Documentation

### 16.2.1 typedef void( TSWITCHEP)(void \*ctx, int x, int y)

typedef for the definition of neighbor function arguments

## 16.3 Function Documentation

### 16.3.1 MB\_errcode MB\_Basins ( MB\_Image \* src, MB\_Image \* marker, Uint32 max\_level, enum MB\_grid\_t grid )

Performs a watershed segmentation of the image using the marker image as a starting point for the flooding. The function returns the catchment basins of the watershed but no actual watershed line. It is recommended to use this functions rather than MB\_Watershed if you are only interested in catchment basins (faster). The result is put into a the 32-bits marker image.

The segmentation is coded as follows into the 32 bit values. | 0 | 1 | 2 | 3 | |-----|-----|-----|-----| | Segment label | Unused | Each byte can be accessed using the function MB\_CopyBytePlane.

### Parameters

- src* the greyscale image to be segmented
- marker* the marker image in which the result of segmentation will be put
- max\_level* the maximum level reached by the water.
- grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

## 17 MB\_BinHitOrMiss.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- MB\_errcode MB\_BinHitOrMiss (MB\_Image \*src, MB\_Image \*dest, Uint32 es0, Uint32 es1, enum MB\_grid\_t grid)

## 17.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-15-2007

## 17.2 Function Documentation

**17.2.1** `MB_errcode MB_BinHitOrMiss ( MB_Image * src, MB_Image * dest, Uint32 es0, Uint32 es1, enum MB_grid_t grid )`

Performs a binary Hit-or-Miss operation on image *imIn* using the structuring elements *es0* and *es1*. Structuring elements are integer values coding which direction must be taken into account. *es0* indicating which neighbor of the current pixel will be checked for 0 value. *es1* those which will be evaluated for 1 value.

For example, in hexagonal grid, it means that if you want to look for a pattern where the neighbors in direction 6 and 1 are true while the current pixel is false just as neighbors 2 and 5, you will encode this in the elements *es0* and *es1* like this : 1 1 0 0 0 X X *es0* = 1+4+32 *es1* = 64+2

### Parameters

- src* output image
- dest* input image (must be different of *src*)
- es0* structuring element for 0 value.
- es1* structuring element for 1 value.
- grid* grid configuration

### Returns

An error code (NO\_ERR if successful)

## 18 MB\_BldDirection.c File Reference

### Typedefs

- typedef void( TSWITCHEP )(PLINE \*plines\_inout, Uint32 linoff\_inout, PLINE \*plines\_mask, Uint32 linoff\_mask, Uint32 bytes\_in, Uint32 nb\_lines, Uint64 \*p\_volume)

### 18.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

1-5-2010

### 18.2 Typedef Documentation

**18.2.1** `typedef void( TSWITCHEP)(PLINE *plines_inout, Uint32 linoff_inout, PLINE *plines_mask, Uint32 linoff_mask, Uint32 bytes_in, Uint32 nb_lines, Uint64 *p_volume)`

typedef for the definition of function arguments

## 19 MB\_BldNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

## Functions

- **MB\_errcode MB\_BldNb32** (**MB\_Image** \*mask, **MB\_Image** \*srctest, **Uint32** dirnum, **Uint64** \*pVolume, enum **MB\_grid\_t** grid)

### 19.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

06-09-2010

### 19.2 Function Documentation

- 19.2.1 MB\_errcode MB\_BldNb32** ( **MB\_Image** \* *mask*, **MB\_Image** \* *srctest*, **Uint32** *dirnum*, **Uint64** \* *pVolume*, enum **MB\_grid\_t** *grid* )

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

#### Parameters

- mask* the mask image
- srctest* the rebuild image
- dirnum* the direction number
- pVolume* the computed volume of the output image
- grid* the grid used (either square or hexagonal)

#### Returns

An error code (`NO_ERR` if successful)

## 20 MB\_BldNb8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

## Functions

- **MB\_errcode MB\_BldNb8** (**MB\_Image** \*mask, **MB\_Image** \*srctest, **Uint32** dirnum, **Uint64** \*pVolume, enum **MB\_grid\_t** grid)

### 20.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-16-2008

## 20.2 Function Documentation

**20.2.1** `MB_errcode MB_BldNb8 ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

## 21 MB\_BldNbb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

### Functions

- `MB_errcode MB_BldNbb (MB_Image *mask, MB_Image *srcdest, Uint32 dirnum, Uint64 *pVolume, enum MB_grid_t grid)`

### 21.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-16-2008

### 21.2 Function Documentation

**21.2.1** `MB_errcode MB_BldNbb ( MB_Image * mask, MB_Image * srcdest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image

*grid* the grid used (either square or hexagonal)

### Returns

An error code (NO\_ERR if successful)

## 22 MB\_Check.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Check (MB_Image *src, Uint32 *isEmpty)`

#### 22.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

29-1-2009

#### 22.2 Function Documentation

##### 22.2.1 `MB_errcode MB_Check ( MB_Image * src, Uint32 * isEmpty )`

Verifies that the image is not empty (all pixels to 0).

##### Parameters

*src* the source image

*isEmpty* an integer which is set to 1 if empty or 0 if not

##### Returns

An error code (NO\_ERR if successful)

## 23 MB\_Compare.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Compare (MB_Image *src, MB_Image *cmp, MB_Image *dest, Sint32 *px, Sint32 *py)`

#### 23.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

6-6-2008

## 23.2 Function Documentation

**23.2.1** `MB_errcode MB_Compare ( MB_Image * src, MB_Image * cmp, MB_Image * dest, Sint32 * px, Sint32 * py )`

Performs a comparison between a source image and a given base image.

### Parameters

*src* the source image

*cmp* the image to which the source image is compared

*dest* destination image

*px* position in x of the first different pixel between the two images (-1 if images are similar)

*py* position in y of the first different pixel between the two images (-1 if images are similar)

### Returns

An error code (NO\_ERR if successful)

## 24 MB\_ConAdd.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_ConAdd (MB_Image *src, Sint32 value, MB_Image *dest)`

### 24.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-25-2007

### 24.2 Function Documentation

**24.2.1** `MB_errcode MB_ConAdd ( MB_Image * src, Sint32 value, MB_Image * dest )`

Adds a constant value to the pixels of an image.

#### Parameters

*src* the source image

*value* the constant value to be added to the pixels

*dest* the image resulting of the addition of image 1 and value.

#### Returns

An error code (NO\_ERR if successful)

## 25 MB\_ConDiv.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- `MB_errcode MB_ConDiv (MB_Image *src, Uint32 value, MB_Image *dest)`

### 25.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 25.2 Function Documentation

#### 25.2.1 `MB_errcode MB_ConDiv ( MB_Image * src, Uint32 value, MB_Image * dest )`

Divides (quotient) the pixels of an image by a constant value.

#### Parameters

*src* the source image

*value* the constant value used in the division

*dest* the image resulting of the division of image 1 by the value.

#### Returns

An error code (NO\_ERR if successful)

## 26 MB\_ConMul.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- `MB_errcode MB_ConMul (MB_Image *src, Uint32 value, MB_Image *dest)`

### 26.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 26.2 Function Documentation

#### 26.2.1 `MB_errcode MB_ConMul ( MB_Image * src, Uint32 value, MB_Image * dest )`

Multiplies a constant value to the pixels of an image.

#### Parameters

*src* the source image

*value* the constant value to be multiplied to the pixels

*dest* the image resulting of the multiplication of image 1 by the value.

### Returns

An error code (NO\_ERR if successful)

## 27 MB\_ConSet.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- **MB\_errcode MB\_ConSet** (MB\_Image \*dest, Uint32 value)

#### 27.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

11-29-2007

#### 27.2 Function Documentation

##### 27.2.1 MB\_errcode MB\_ConSet ( MB\_Image \* dest, Uint32 value )

Fills an image with a specific value

##### Parameters

*dest* the image

*value* the value to fill the image

##### Returns

An error code (NO\_ERR if successful)

## 28 MB\_ConSub.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- **MB\_errcode MB\_ConSub** (MB\_Image \*src, Sint32 value, MB\_Image \*dest)

#### 28.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

6-13-2007

## 28.2 Function Documentation

### 28.2.1 `MB_errcode MB_ConSub ( MB_Image * src, Sint32 value, MB_Image * dest )`

Subtracts a constant value to the pixels of an image.

#### Parameters

*src* the source image

*value* the constant value to be subtracted to the pixels

*dest* the image resulting of the subtraction of image 1 and value.

#### Returns

An error code (NO\_ERR if successful)

## 29 MB\_Convert.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Convert1to8 (MB_Image *src, MB_Image *dest)`
- `MB_errcode MB_Convert8to1 (MB_Image *src, MB_Image *dest)`
- `MB_errcode MB_Convert (MB_Image *src, MB_Image *dest)`

### 29.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

5-29-2008

### 29.2 Function Documentation

#### 29.2.1 `MB_errcode MB_Convert ( MB_Image * src, MB_Image * dest )`

Converts an image of a given depth into another depth

This function does not depend on the window computation currently set.

#### Parameters

*src* source image

*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

### 29.2.2 `MB_errcode MB_Convert1to8 ( MB_Image * src, MB_Image * dest )`

Converts a binary image to an 8-bit image. Pixels to True are set to 255 and to 0 otherwise

#### Parameters

*src* source image  
*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

### 29.2.3 `MB_errcode MB_Convert8to1 ( MB_Image * src, MB_Image * dest )`

Converts an 8-bit image to a binary image. Pixels at 255 are set to True and to False otherwise

#### Parameters

*src* source image  
*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

## 30 MB\_Copy.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Copy (MB_Image *src, MB_Image *dest)`
- `MB_errcode MB_CopyLine (MB_Image *src, MB_Image *dest, Uint32 insrc_pos, Uint32 indest_pos)`
- `MB_errcode MB_CropCopy (MB_Image *src, Uint32 x_src, Uint32 y_src, MB_Image *dest, Uint32 x_dest, Uint32 y_dest, Uint32 w, Uint32 h)`

### 30.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

13-05-2010

### 30.2 Function Documentation

#### 30.2.1 `MB_errcode MB_Copy ( MB_Image * src, MB_Image * dest )`

Copies an image data contents into another image This copy works with same size images.

#### Parameters

*src* the source image

*dest* the destination image

#### Returns

An error code (NO\_ERR if successful)

**30.2.2** `MB_errcode MB_CopyLine ( MB_Image * src, MB_Image * dest, Uint32 insrc_pos, Uint32 indest_pos )`

Copies an image line contents into another image line

#### Parameters

*src* the source image

*dest* the destination image

*insrc\_pos* the position of the line copied from src

*indest\_pos* the position in dest in which the line is copied

#### Returns

An error code (NO\_ERR if successful)

**30.2.3** `MB_errcode MB_CropCopy ( MB_Image * src, Uint32 x_src, Uint32 y_src, MB_Image * dest, Uint32 x_dest, Uint32 y_dest, Uint32 w, Uint32 h )`

Copies an image data contents into another image. This copy can work with image of different sizes. As the size can be different the position where the copy occurs must be specified for both images as well as the size of the copy. The function will compute the actual crop inside the source and destination images. Works only with non binary images.

#### Parameters

*src* the source image

*x\_src* the x position in the source image where the copy should begin

*y\_src* the y position in the source image where the copy should begin

*dest* the destination image

*x\_dest* the x position in the destination image where the copy should happen

*y\_dest* the y position in the destination image where the copy should happen

*w* the width of the copy

*h* the height of the copy

#### Returns

An error code (NO\_ERR if successful)

## 31 MB\_CopyBitPlane.c File Reference

```
#include "mambaApi_loc.h"
```

#### Functions

- `Uint8 SET_BIT_PLANE (Uint8 value, Uint8 bitval, Uint32 pos)`
- `Uint8 EXTRACT_BIT_PLANE (Uint8 value, Uint32 pos)`
- `MB_errcode MB_CopyBitPlane (MB_Image *src, MB_Image *dest, Uint32 plane)`

## 31.1 Detailed Description

### Author

Nicolas Beucher

### Date

5-29-2008

## 31.2 Function Documentation

### 31.2.1 `Uint8 EXTRACT_BIT_PLANE ( Uint8 value, Uint32 pos )`

Extracts the bit at the given position.

#### Parameters

*value* the value in which the bit will be extracted

*pos* the position of the bit

#### Returns

the value of the bit

### 31.2.2 `MB_errcode MB_CopyBitPlane ( MB_Image * src, MB_Image * dest, Uint32 plane )`

Inserts or extracts the bit plane in/out an image src into dest.

#### Parameters

*src* source image

*dest* destination image

*plane* the plane number

#### Returns

An error code (NO\_ERR if successful)

### 31.2.3 `Uint8 SET_BIT_PLANE ( Uint8 value, Uint8 bitval, Uint32 pos )`

Sets the bit at the given position.

#### Parameters

*value* the value in which the bit will be modified

*bitval* the bit value we want to set

*pos* the position of the bit

#### Returns

the modified value

## 32 MB\_CopyBytePlane.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- `MB_errcode MB_CopyBytePlane (MB_Image *src, MB_Image *dest, Uint32 plane)`

### 32.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

5-29-2008

### 32.2 Function Documentation

#### 32.2.1 `MB_errcode MB_CopyBytePlane ( MB_Image * src, MB_Image * dest, Uint32 plane )`

Inserts or extracts the byte plane in/out an image *src* into/from *dest*.

#### Parameters

*src* source image

*dest* destination image

*plane* the plane number

#### Returns

An error code (NO\_ERR if successful)

## 33 MB\_Create.c File Reference

```
#include "mambaApi_loc.h"
```

### Defines

- `#define MB_ROUND_W 64`
- `#define MB_ROUND_H 2`
- `#define MB_MAX_IMAGE_W_SIZE 4096`
- `#define MB_MAX_IMAGE_H_SIZE MB_MAX_IMAGE_W_SIZE`

### Functions

- `MB_errcode MB_Create (MB_Image *image, Uint32 width, Uint32 height, Uint32 depth)`
- `MB_errcode MB_Destroy (MB_Image *image)`

### Variables

- `Uint32 MB_refcounter = 0`

### 33.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

5-27-2008

### 33.2 Define Documentation

#### 33.2.1 `#define MB_MAX_IMAGE_H_SIZE MB_MAX_IMAGE_W_SIZE`

Image limit size in Y

#### 33.2.2 `#define MB_MAX_IMAGE_W_SIZE 4096`

Image limit size in x

#### 33.2.3 `#define MB_ROUND_H 2`

Making sure the image size is multiple of 2 for the heighth

#### 33.2.4 `#define MB_ROUND_W 64`

Making sure the image size is multiple of 64 for the width

### 33.3 Function Documentation

#### 33.3.1 `MB_errcode MB_Create ( MB_Image * image, Uint32 width, Uint32 height, Uint32 depth )`

Creates an image (memory allocation) with the correct size and depth given as argument. The size is deduced from the requested size given in argument. The size must be a multiple of MB\_ROUND\_W for width and MB\_ROUND\_H for height. The size cannot be greater than MB\_MAX\_IMAGE\_W\_SIZExMB\_MAX\_IMAGE\_H\_SIZE.

#### Parameters

*image* the created image  
*width* the width of the created image  
*height* the height of the created image  
*depth* the depth of the created image

#### Returns

An error code (NO\_ERR if successful)

#### 33.3.2 `MB_errcode MB_Destroy ( MB_Image * image )`

Destroys an image (memory freeing)

#### Parameters

*image* the image to be destroyed

## Returns

An error code (NO\_ERR if successful)

## 33.4 Variable Documentation

### 33.4.1 Uint32 MB\_refcounter = 0

image counter

## 34 MB\_Diff.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- **MB\_errcode MB\_Diff (MB\_Image \*src1, MB\_Image \*src2, MB\_Image \*dest)**

### 34.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-29-2007

### 34.2 Function Documentation

#### 34.2.1 MB\_errcode MB\_Diff ( MB\_Image \* src1, MB\_Image \* src2, MB\_Image \* dest )

Computes the set difference between two images. The result image pixel value is the pixel value of image 1 if this value was greater than value of pixel 2 otherwise the minimum possible value is set for the pixel

#### Parameters

*src1* source image 1

*src2* source image 2

*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

## 35 MB\_DiffNb8.c File Reference

```
#include "mambaApi_loc.h"
```

```
#include "MB_Neighbor.c"
```

## Defines

- **#define EDGE\_TYPE Uint32**

## Functions

- `MB_errcode MB_DiffNb8 (MB_Image *src, MB_Image *srcdest, Uint32 nbrnum, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 35.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 35.2 Define Documentation

#### 35.2.1 `#define EDGE_TYPE Uint32`

Data type of the value used to represent the edge

### 35.3 Function Documentation

#### 35.3.1 `MB_errcode MB_DiffNb8 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Computes the set difference between two greyscale image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see `MB_ngh.h`).

#### Parameters

- src* source image in which the neighbors are taken
- srcdest* source of the central pixel and destination image
- nbrnum* the neighbor index
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (`NO_ERR` if successful)

## 36 MB\_DiffNbb.c File Reference

```
#include "mambaApi_loc.h"
```

```
#include "MB_Neighbor.c"
```

### Defines

- `#define EDGE_TYPE binaryT`

### Functions

- `MB_errcode MB_DiffNbb (MB_Image *src, MB_Image *srcdest, Uint32 nbrnum, enum MB_grid_t grid, enum MB_edgemode_t edge)`

## 36.1 Detailed Description

### Author

Nicolas Beucher

### Date

6-18-2008

## 36.2 Define Documentation

### 36.2.1 #define EDGE\_TYPE binaryT

Data type of the value used to represent the edge

## 36.3 Function Documentation

### 36.3.1 MB\_errcode MB\_DiffNbb ( MB\_Image \* *src*, MB\_Image \* *srcdest*, Uint32 *nbrnum*, enum MB\_grid\_t *grid*, enum MB\_edgemode\_t *edge* )

Computes the set difference between two binary image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

- src* source image in which the neighbors are taken
- srcdest* source of the central pixel and destination image
- nbrnum* the neighbor index
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

## 37 MB\_Distanceb.c File Reference

```
#include "mambaApi_loc.h"
```

### Defines

- #define IS\_SET(pix) ((pix)!=0)
- #define VAL(p\_pix) (\*p\_pix)
- #define LEFT(p\_pix) (\*(p\_pix-1))
- #define RIGHT(p\_pix) (\*(p\_pix+1))
- #define EDGE\_DIST 0x00010000

### Typedefs

- typedef void( TSWITCHEP )(PLINE \*plines\_out, Uint32 linoff\_out, PLINE \*plines\_in, Uint32 linoff\_in, Uint32 bytes\_in, Uint32 bytes\_out, Uint32 nb\_lines, binaryT edge\_val)

## Functions

- `MB_errcode MB_Distanceb (MB_Image *src, MB_Image *dest, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 37.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

12-27-2008

### 37.2 Define Documentation

#### 37.2.1 `#define EDGE_DIST 0x00010000`

When a filled edge is selected, we must ensure that pixels touching the edge will be set to an 'infinite' distance, i.e. greater than every possible distance. As the image size is limited to 4100x4100, we select a greater value than this to emulate the edge infinite distance

<- 65536 should be ok even if the image size is extended

#### 37.2.2 `#define IS_SET( pix ) ((pix)!=0)`

pixel value defines

#### 37.2.3 `#define LEFT( p_pix ) (*(p_pix-1))`

Macro returning the value of the left neighbor pixel

#### 37.2.4 `#define RIGHT( p_pix ) (*(p_pix+1))`

Macro returning the value of the right neighbor pixel

#### 37.2.5 `#define VAL( p_pix ) (*p_pix)`

Macro returning the value of a pixel

### 37.3 Typedef Documentation

#### 37.3.1 `typedef void( TSWITCHEP)(PLINE *plines_out, Uint32 linoff_out, PLINE *plines_in, Uint32 linoff_in, Uint32 bytes_in, Uint32 bytes_out, Uint32 nb_lines, binaryT edge_val)`

typedef for the definition of function arguments

### 37.4 Function Documentation

#### 37.4.1 `MB_errcode MB_Distanceb ( MB_Image * src, MB_Image * dest, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Computes for each pixel the distance to the edge of the set in which the pixel is found

The algorithm works with two passes, first from the top left to the bottom right and then back.

### Parameters

- src* the binary source image
- dest* the 32-bits image in which the distance for each pixel is stored
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

## 38 MB\_DualBldNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

### Functions

- **MB\_errcode MB\_DualBldNb32** (**MB\_Image** \*mask, **MB\_Image** \*srctest, **Uint32** dirnum, **Uint64** \*pVolume, enum **MB\_grid\_t** grid)

### 38.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

06-09-2010

### 38.2 Function Documentation

**38.2.1 MB\_errcode MB\_DualBldNb32** ( **MB\_Image** \* *mask*, **MB\_Image** \* *srctest*, **Uint32** *dirnum*, **Uint64** \* *pVolume*, enum **MB\_grid\_t** *grid* )

(re)Builds (dual operation) an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

- mask* the mask image
- srctest* the rebuild image
- dirnum* the direction number
- pVolume* the computed volume of the output image
- grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

## 39 MB\_DualBldNb8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

### Functions

- **MB\_errcode MB\_DualBldNb8** (**MB\_Image** \*mask, **MB\_Image** \*srctest, **Uint32** dirnum, **Uint64** \*pVolume, enum **MB\_grid\_t** grid)

### 39.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-16-2008

### 39.2 Function Documentation

**39.2.1 MB\_errcode MB\_DualBldNb8** ( **MB\_Image** \* *mask*, **MB\_Image** \* *srctest*, **Uint32** *dirnum*, **Uint64** \* *pVolume*, enum **MB\_grid\_t** *grid* )

(re)Builds (dual operation) an image according to a direction and a mask image. The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*mask* the mask image  
*srctest* the rebuild image  
*dirnum* the direction number  
*pVolume* the computed volume of the output image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

## 40 MB\_DualBldNbb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_BldDirection.c"
```

### Functions

- **MB\_errcode MB\_DualBldNbb** (**MB\_Image** \*mask, **MB\_Image** \*srctest, **Uint32** dirnum, **Uint64** \*pVolume, enum **MB\_grid\_t** grid)

## 40.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-16-2008

## 40.2 Function Documentation

### 40.2.1 `MB_errcode MB_DualBldNbb ( MB_Image * mask, MB_Image * srctest, Uint32 dirnum, Uint64 * pVolume, enum MB_grid_t grid )`

(re)Builds an image according to a direction and a mask image. The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

#### Parameters

- mask* the mask image
- srctest* the rebuild image
- dirnum* the direction number
- pVolume* the computed volume of the output image
- grid* the grid used (either square or hexagonal)

#### Returns

An error code (`NO_ERR` if successful)

## 41 MB\_Error.c File Reference

```
#include "MB_error.h"
```

### Functions

- `char * MB_StrErr (MB_errcode error_nb)`

### Variables

- `char * err_str []`

## 41.1 Detailed Description

### Author

Nicolas Beucher

### Date

6-4-2008

## 41.2 Function Documentation

### 41.2.1 `char* MB_StrErr ( MB_errcode error_nb )`

Returns an explanation of the error code

#### Parameters

*error\_nb* the error code number

## 41.3 Variable Documentation

### 41.3.1 `char* err_str[]`

Initial value:

```
{
  "No error",
  "Incompatible image sizes",
  "Incorrect image depth",
  "Bad parameter",
  "Bad value",
  "Incorrect direction or neighbor for given grid",
  "Memory allocation impossible",
  "Incorrect image dimensions",
  "Incorrect load data (size or type)",
  "Unknown Error"
}
```

Error value interpretation

## 42 MB\_error.h File Reference

### Enumerations

- enum MB\_errcode {
   
NO\_ERR, ERR\_BAD\_SIZE, ERR\_BAD\_DEPTH, ERR\_BAD\_PARAMETER,
   
ERR\_BAD\_VALUE, ERR\_BAD\_DIRECTION, ERR\_CANT\_ALLOCATE\_
   
MEMORY, ERR\_BAD\_IMAGE\_DIMENSIONS,
   
ERR\_LOAD\_DATA, ERR\_UNKNOWN }

### Functions

- `char * MB_StrErr (MB_errcode error_nb)`

### 42.1 Detailed Description

#### Date

11-4-2007

This file contains the complete list of error codes returned by the API functions.

### 42.2 Enumeration Type Documentation

#### 42.2.1 enum MB\_errcode

Type definition for error code

**Enumerator:**

***NO\_ERR*** Value returned by function when no error was encountered.

***ERR\_BAD\_SIZE*** Value returned by function when an error of size inside the image was encountered. For example if the width of a first image was not corresponding with the width of a second image when both are used in the same computations.

***ERR\_BAD\_DEPTH*** Value returned by function when an error of depth inside the image was encountered. For example, the function expected 8-bit pixels and got 1-bit pixels.

***ERR\_BAD\_PARAMETER*** Value returned by function when a given parameter was incorrect.

***ERR\_BAD\_VALUE*** Value returned by function when a given value (as argument) was incorrect.

***ERR\_BAD\_DIRECTION*** Value returned when a function requiring a direction (shift, neighbor) is given an incorrect argument for direction (allowed value depends on the grid).

***ERR\_CANT\_ALLOCATE\_MEMORY*** Value returned when the allocation for image memory failed.

***ERR\_BAD\_IMAGE\_DIMENSIONS*** Value returned when the dimension of the image given in argument of a function is incorrect.

***ERR\_LOAD\_DATA*** Value returned when the data given to a load function is incorrect (size or type)

***ERR\_UNKNOWN*** Value never returned (reserved for the error function)

## 42.3 Function Documentation

### 42.3.1 `char* MB_StrErr ( MB_errcode error_nb )`

Returns an explanation of the error code

**Parameters**

*error\_nb* the error code number

## 43 MB\_Frame.c File Reference

```
#include "mambaApi_loc.h"
```

**Functions**

- `MB_errcode MB_Frame (MB_Image *src, Uint32 thresval, Uint32 *ulx, Uint32 *uly, Uint32 *brx, Uint32 *bry)`

### 43.1 Detailed Description

**Author**

Nicolas Beucher

**Date**

3-3-2009

### 43.2 Function Documentation

#### 43.2.1 `MB_errcode MB_Frame ( MB_Image * src, Uint32 thresval, Uint32 * ulx, Uint32 * uly, Uint32 * brx, Uint32 * bry )`

Returns the smallest frame that contains all the pixels of image that are greater or equal to the given threshold value, using the four last pointers to describe it

## Parameters

- src* source image
- thresval* the threshold value used to compute the frame
- ulx* the x-coordinate of the upper left corner of the frame
- uly* the y-coordinate of the upper left corner of the frame
- brx* the x-coordinate of the bottom right corner of the frame
- bry* the y-coordinate of the bottom right corner of the frame

## Returns

An error code (NO\_ERR if successful)

## 44 MB\_HierarBld.c File Reference

```
#include "mambaApi_loc.h"
```

### Data Structures

- struct MB\_Hierarbld\_Ctx

### Typedefs

- typedef void( TSWITCHEP )(void \*ctx, int x, int y)

### Functions

- MB\_errcode MB\_HierarBld (MB\_Image \*mask, MB\_Image \*srctest, enum MB\_grid\_t grid)

#### 44.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

19-08-2010

#### 44.2 Typedef Documentation

##### 44.2.1 typedef void( TSWITCHEP)(void \*ctx, int x, int y)

typedef for the definition of neighbor function arguments

#### 44.3 Function Documentation

##### 44.3.1 MB\_errcode MB\_HierarBld ( MB\_Image \* mask, MB\_Image \* srctest, enum MB\_grid\_t grid )

(re)Builds an image according to a mask image and using a hierarchical list to compute the rebuild.

##### Parameters

- mask* the mask image

*srcdest* the rebuild image  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

## 45 MB\_HierarDualBld.c File Reference

```
#include "mambaApi_loc.h"
```

### Data Structures

- struct MB\_Hierardualbld\_Ctx

### Typedefs

- typedef void( TSWITCHEP )(void \*ctx, int x, int y)

### Functions

- MB\_errcode MB\_HierarDualBld (MB\_Image \*mask, MB\_Image \*srcdest, enum MB\_grid\_t grid)

#### 45.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

19-08-2010

#### 45.2 Typedef Documentation

##### 45.2.1 typedef void( TSWITCHEP )(void \*ctx, int x, int y)

typedef for the definition of neighbor function arguments

#### 45.3 Function Documentation

##### 45.3.1 MB\_errcode MB\_HierarDualBld ( MB\_Image \* mask, MB\_Image \* srcdest, enum MB\_grid\_t grid )

(re)Builds (dual operation) an image according to a mask image and using a hierarchical list to compute the rebuild.

##### Parameters

*mask* the mask image  
*srcdest* the rebuild image  
*grid* the grid used (either square or hexagonal)

##### Returns

An error code (NO\_ERR if successful)

## 46 MB\_Histo.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Histo (MB_Image *src, Uint32 *phisto)`

#### 46.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

11-25-2007

#### 46.2 Function Documentation

##### 46.2.1 `MB_errcode MB_Histo ( MB_Image * src, Uint32 * phisto )`

Computes the histogram of an image. The histogram is an array with a minimal size of 256.

##### Parameters

*src* source image

*phisto* pointer to the histogram array

##### Returns

An error code (NO\_ERR if successful)

## 47 MB\_Inf.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Inf (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

#### 47.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

11-25-2007

#### 47.2 Function Documentation

##### 47.2.1 `MB_errcode MB_Inf ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Determines the inferior value between the pixels of two images. The result is put in the corresponding pixel position in the destination image.

### Parameters

*src1* image 1  
*src2* image 2  
*dest* destination image

### Returns

An error code (NO\_ERR if successful)

## 48 MB\_InfFarNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

### Defines

- #define EDGE\_TYPE PIX32

### Functions

- MB\_errcode MB\_InfFarNb32 (MB\_Image \*src, MB\_Image \*srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge)

### 48.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

29-6-2010

### 48.2 Define Documentation

#### 48.2.1 #define EDGE\_TYPE PIX32

Data type of the value used to represent the edge

### 48.3 Function Documentation

#### 48.3.1 MB\_errcode MB\_InfFarNb32 ( MB\_Image \* src, MB\_Image \* srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge )

Looks for the minimum between two 32-bits image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)

*grid* the grid used (either square or hexagonal)

*edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

## 49 MB\_InfFarNb8.c File Reference

```
#include "mambaApi_loc.h"
```

```
#include "MB_ShftDirection.c"
```

### Defines

- #define EDGE\_TYPE Uint32

### Functions

- MB\_errcode MB\_InfFarNb8 (MB\_Image \*src, MB\_Image \*srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge)

#### 49.1 Detailed Description

##### Author

Nicolas Beucher

##### Date

25-6-2010

#### 49.2 Define Documentation

##### 49.2.1 #define EDGE\_TYPE Uint32

Data type of the value used to represent the edge

#### 49.3 Function Documentation

##### 49.3.1 MB\_errcode MB\_InfFarNb8 ( MB\_Image \* src, MB\_Image \* srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge )

Looks for the minimum between two greyscale image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

##### Parameters

*src* source image in which the neighbor are taken

*srcdest* source of the central pixel and destination image

*nbrnum* the neighbor index

*count* the amplitude of the shift (in pixels)

*grid* the grid used (either square or hexagonal)

*edge* the kind of edge to use (behavior for pixel near edge depends on it)

## Returns

An error code (NO\_ERR if successful)

## 50 MB\_InfFarNbb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

## Defines

- #define **EDGE\_TYPE** binaryT

## Functions

- **MB\_errcode** MB\_InfFarNbb (MB\_Image \*src, MB\_Image \*srctest, **Uint32** nbrnum, **Uint32** count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge)

### 50.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

24-6-2010

### 50.2 Define Documentation

#### 50.2.1 #define EDGE\_TYPE binaryT

Data type of the value used to represent the edge

### 50.3 Function Documentation

#### 50.3.1 **MB\_errcode** MB\_InfFarNbb ( MB\_Image \* *src*, MB\_Image \* *srctest*, **Uint32** *nbrnum*, **Uint32** *count*, enum MB\_grid\_t *grid*, enum MB\_edgemode\_t *edge* )

Looks for the minimum between two binary image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

- src* source image in which the neighbor are taken
- srctest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* the amplitude of the shift (in pixels)
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

## 51 MB\_InfNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

### Defines

- `#define EDGE_TYPE_PIX32`

### Functions

- `MB_errcode MB_InfNb32 (MB_Image *src, MB_Image *srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 51.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 51.2 Define Documentation

#### 51.2.1 `#define EDGE_TYPE_PIX32`

Data type of the value used to represent the edge

### 51.3 Function Documentation

#### 51.3.1 `MB_errcode MB_InfNb32 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two 32-bits image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see `MB_ngh.h`).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* if *src* and *srcdest* are the same, the operation is repeated count times  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (`NO_ERR` if successful)

## 52 MB\_InfNb8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

## Defines

- `#define EDGE_TYPE Uint32`

## Functions

- `MB_errcode MB_InfNb8 (MB_Image *src, MB_Image *srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 52.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 52.2 Define Documentation

#### 52.2.1 `#define EDGE_TYPE Uint32`

Data type of the value used to represent the edge

### 52.3 Function Documentation

#### 52.3.1 `MB_errcode MB_InfNb8 ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two greyscale image pixels (a central pixel and its neighbor in the other image)  
The neighbor depends on the grid used (see `MB_ngh.h`).

#### Parameters

- *src* source image in which the neighbor are taken
- *srctest* source of the central pixel and destination image
- *nbrnum* the neighbor index
- *count* if *src* and *srctest* are the same, the operation is repeated count times
- *grid* the grid used (either square or hexagonal)
- *edge* the kind of edge to use (behavior for pixel near edge depends on it)

#### Returns

An error code (`NO_ERR` if successful)

## 53 MB\_InfNb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

## Defines

- `#define EDGE_TYPE binaryT`

## Functions

- `MB_errcode MB_InfNbb (MB_Image *src, MB_Image *srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 53.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 53.2 Define Documentation

#### 53.2.1 `#define EDGE_TYPE binaryT`

Data type of the value used to represent the edge

### 53.3 Function Documentation

#### 53.3.1 `MB_errcode MB_InfNbb ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the minimum between two binary image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see `MB_ngh.h`).

#### Parameters

- src* source image in which the neighbor are taken
- srctest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* if *src* and *srctest* are the same, the operation is repeated count times
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (`NO_ERR` if successful)

## 54 MB\_Inv.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Inv (MB_Image *src, MB_Image *dest)`

## 54.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-25-2007

## 54.2 Function Documentation

### 54.2.1 `MB_errcode MB_Inv ( MB_Image * src, MB_Image * dest )`

Inverts the pixels values (negation) of the source image.

#### Parameters

*src* source image

*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

## 55 MB\_Labelb.c File Reference

```
#include "mambaApi_loc.h"
```

### Data Structures

- struct `MB_Label`

### Defines

- #define `VAL(p_pix)` (\*p\_pix)
- #define `LEFT(p_pix)` (\*(p\_pix-1))
- #define `RIGHT(p_pix)` (\*(p\_pix+1))

### Typedefs

- typedef void( `TSWITCHEP` )( `PLINE *plines_out, Uint32 linoff_out, PLINE *plines_in, Uint32 linoff_in, Uint32 bytes_in, Uint32 nb_lines, MB_Label *labels` )

### Functions

- `MB_errcode MB_Labelb (MB_Image *src, MB_Image *dest, Uint32 lblow, Uint32 lblhigh, Uint32 *pNbobj, enum MB_grid_t grid)`

## 55.1 Detailed Description

### Author

Nicolas Beucher

### Date

12-27-2008

## 55.2 Define Documentation

### 55.2.1 #define LEFT( *p\_pix* ) (\*(p\_pix-1))

Macro returning the value of the left neighbor pixel

### 55.2.2 #define RIGHT( *p\_pix* ) (\*(p\_pix+1))

Macro returning the value of the right neighbor pixel

### 55.2.3 #define VAL( *p\_pix* ) (\*p\_pix)

Macro returning the value of a pixel

## 55.3 Typedef Documentation

### 55.3.1 typedef void( TSWITCHEP)(PLINE \*plines\_out, Uint32 linoff\_out, PLINE \*plines\_in, Uint32 linoff\_in, Uint32 bytes\_in, Uint32 nb\_lines, MB\_Label \*labels)

typedef for the definition of function arguments

## 55.4 Function Documentation

### 55.4.1 MB\_errcode MB\_Labelb ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 *lblow*, Uint32 *lbhigh*, Uint32 \* *pNbobj*, enum MB\_grid\_t *grid* )

Labeling the object found in *src* image.

#### Parameters

- src* the binary source image where the object must be labelled
- dest* the 32-bit image where object are labelled
- lblow* the lowest value allowed for label on the low byte (must be inferior to *lbhigh*)
- lbhigh* the first high value NOT allowed for label on the low byte (maximum allowed is 256)
- pNbobj* the number of object founds
- grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

## 56 MB\_LoadExtract.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- MB\_errcode MB\_Load8 (MB\_Image \*image, PIX8 \*indata, Uint32 len)
- MB\_errcode MB\_Load32 (MB\_Image \*image, PIX8 \*indata, Uint32 len)
- MB\_errcode MB\_Load (MB\_Image \*image, PIX8 \*indata, Uint32 len)
- MB\_errcode MB\_Extract8 (MB\_Image \*image, PIX8 \*\*outdata, Uint32 \*len)
- MB\_errcode MB\_Extract32 (MB\_Image \*image, PIX8 \*\*outdata, Uint32 \*len)
- MB\_errcode MB\_Extract (MB\_Image \*image, PIX8 \*\*outdata, Uint32 \*len)

## 56.1 Detailed Description

### Author

Nicolas Beucher

### Date

5-28-2008

## 56.2 Function Documentation

### 56.2.1 `MB_errcode MB_Extract ( MB_Image * image, PIX8 ** outdata, Uint32 * len )`

Reads an image data contents and put it in an array

#### Parameters

*image* the image to read

*outdata* pointer to the array created (malloc) and filled with the pixel data of the image

*len* the length in bytes of data extracted

#### Returns

An error code (NO\_ERR if successful)

### 56.2.2 `MB_errcode MB_Extract32 ( MB_Image * image, PIX8 ** outdata, Uint32 * len )`

Reads a 32-bits image data contents and put it in an array

#### Parameters

*image* the image to read

*outdata* pointer to the array created (malloc) and filled with the pixel data of the image

*len* the length in bytes of data extracted

#### Returns

An error code (NO\_ERR if successful)

### 56.2.3 `MB_errcode MB_Extract8 ( MB_Image * image, PIX8 ** outdata, Uint32 * len )`

Reads a grey scale image data contents and put it in an array

#### Parameters

*image* the image to read

*outdata* pointer to the array created (malloc) and filled with the pixel data of the image

*len* the length in bytes of data extracted

#### Returns

An error code (NO\_ERR if successful)

### 56.2.4 MB\_errcode MB\_Load ( MB\_Image \* *image*, PIX8 \* *indata*, Uint32 *len* )

Loads an image data with data given in argument

#### Parameters

- image* the image to fill
- indata* the data to fill the image with (complete pixels values)
- len* the length of data given

#### Returns

An error code (NO\_ERR if successful)

### 56.2.5 MB\_errcode MB\_Load32 ( MB\_Image \* *image*, PIX8 \* *indata*, Uint32 *len* )

Loads a 32-bits image data with data given in argument

#### Parameters

- image* the image to fill
- indata* the data to fill the image with (complete pixels values)
- len* the length of data given

#### Returns

An error code (NO\_ERR if successful)

### 56.2.6 MB\_errcode MB\_Load8 ( MB\_Image \* *image*, PIX8 \* *indata*, Uint32 *len* )

Loads a grey scale image data with data given in argument

#### Parameters

- image* the image to fill
- indata* the data to fill the image with (complete pixels values)
- len* the length of data given

#### Returns

An error code (NO\_ERR if successful)

## 57 MB\_Lookup.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- MB\_errcode MB\_Lookup (MB\_Image \*src, MB\_Image \*dest, Uint32 \*ptab)

### 57.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-25-2007

## 57.2 Function Documentation

### 57.2.1 MB\_errcode MB\_Lookup ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 \* *ptab* )

Applies the function in the lookup table to the pixels of source image. A pixel value is changed to a new value accordingly with its current value

#### Parameters

*src* source image  
*dest* destination image  
*ptab* the lookup table pointer

#### Returns

An error code (NO\_ERR if successful)

## 58 MB\_Mask.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- MB\_errcode MB\_Mask (MB\_Image \**src*, MB\_Image \**dest*, Uint32 *maskf*, Uint32 *maskt*)

### 58.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

5-29-2008

### 58.2 Function Documentation

#### 58.2.1 MB\_errcode MB\_Mask ( MB\_Image \* *src*, MB\_Image \* *dest*, Uint32 *maskf*, Uint32 *maskt* )

Converts a binary image in a grey scale image (8-bits) or in a 32-bits image using value *maskf* to replace 0 and *maskt* to replace 1.

#### Parameters

*src* binary source image  
*dest* destination image  
*maskf* for 0 (false) pixel value  
*maskt* for 1 (true) pixel value

#### Returns

An error code (NO\_ERR if successful)

## 59 MB\_Mul.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- `MB_errcode MB_Mul (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

### 59.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

2-27-2009

### 59.2 Function Documentation

**59.2.1** `MB_errcode MB_Mul ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Multiplies the pixels of two images and puts the result in the third image. Depending on the format of the target image, the result may be saturated or not. You can perform the following additions : 1-bit \* 1-bit = 1-bit (call the `MB_And` function) 1-bit \* 8-bit = 8-bit 1-bit \* 8-bit = 32-bit 1-bit \* 32-bit = 32-bit 8-bit \* 8-bit = 8-bit (saturated) 8-bit \* 8-bit = 32-bit 8-bit \* 32-bit = 32-bit 32-bit \* 32-bit = 32-bit

#### Parameters

*src1* image 1

*src2* image 2

*dest* image resulting of the multiplication of image 1 and 2.

#### Returns

An error code (`NO_ERR` if successful)

## 60 MB\_Neighbor.c File Reference

### Typedefs

- typedef void( `TSWITCHEP` )( `PLINE` \**plines\_inout*, `UInt32` *linoff\_inout*, `PLINE` \**plines\_in*, `UInt32` *linoff\_in*, `UInt32` *bytes\_in*, `UInt32` *nb\_lines*, `UInt32` *count*, `EDGE_TYPE` *edge\_val*)

### 60.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

1-5-2010

### 60.2 Typedef Documentation

**60.2.1** typedef void( `TSWITCHEP`)( `PLINE` \**plines\_inout*, `UInt32` *linoff\_inout*, `PLINE` \**plines\_in*, `UInt32` *linoff\_in*, `UInt32` *bytes\_in*, `UInt32` *nb\_lines*, `UInt32` *count*, `EDGE_TYPE` *edge\_val*)

typedef for the definition of function arguments

## 61 MB\_Or.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Or (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

### 61.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-1-2008

### 61.2 Function Documentation

#### 61.2.1 `MB_errcode MB_Or ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Applies a bitwise OR on the pixels of two images. All the images must have the same depth for correct work.

#### Parameters

*src1* image 1

*src2* image 2

*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

## 62 MB\_Pixel.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_PutPixel (MB_Image *dest, Uint32 pixVal, Uint32 x, Uint32 y)`
- `MB_errcode MB_GetPixel (MB_Image *src, Uint32 *pixVal, Uint32 x, Uint32 y)`

### 62.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

29-1-2009

## 62.2 Function Documentation

**62.2.1** `MB_errcode MB_GetPixel ( MB_Image * src, Uint32 * pixVal, Uint32 x, Uint32 y )`

Gets the pixel value inside the image at the given position

### Parameters

*src* the image  
*pixVal* the returned pixel value  
*x* position in x of the pixel targeted  
*y* position in y of the pixel targeted

### Returns

An error code (NO\_ERR if successful)

**62.2.2** `MB_errcode MB_PutPixel ( MB_Image * dest, Uint32 pixVal, Uint32 x, Uint32 y )`

Puts the pixel value inside the image at the given position

### Parameters

*dest* the image  
*pixVal* the pixel value  
*x* position in x of the pixel targeted  
*y* position in y of the pixel targeted

### Returns

An error code (NO\_ERR if successful)

## 63 MB\_Range.c File Reference

```
#include "mambaApi_loc.h"
#include <stdint.h>
```

### Functions

- `MB_errcode MB_Range (MB_Image *src, Uint32 *min, Uint32 *max)`
- `MB_errcode MB_depthRange (MB_Image *src, Uint32 *min, Uint32 *max)`

### 63.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

5-29-2008

## 63.2 Function Documentation

### 63.2.1 `MB_errcode MB_depthRange ( MB_Image * src, Uint32 * min, Uint32 * max )`

gives the minimum and maximum possible values of the image pixels given the image depth

#### Parameters

- src* source image
- min* the minimum possible value of the pixels
- max* the maximum possible value of the pixels

#### Returns

An error code (NO\_ERR if successful)

### 63.2.2 `MB_errcode MB_Range ( MB_Image * src, Uint32 * min, Uint32 * max )`

gives the minimum and maximum values of the image pixels i.e its range.

#### Parameters

- src* source image
- min* the minimum value of the pixels
- max* the maximum value of the pixels

#### Returns

An error code (NO\_ERR if successful)

## 64 MB\_ShftDirection.c File Reference

### Typedefs

- typedef void( **TSWITCHEP** )(PLINE \*plines\_out, Uint32 linoff\_out, PLINE \*plines\_in, Uint32 linoff\_in, Uint32 bytes\_in, Sint32 nb\_lines, Sint32 count, EDGE\_TYPE edge\_val)

### 64.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

1-5-2010

### 64.2 Typedef Documentation

#### 64.2.1 `typedef void( TSWITCHEP)(PLINE *plines_out, Uint32 linoff_out, PLINE *plines_in, Uint32 linoff_in, Uint32 bytes_in, Sint32 nb_lines, Sint32 count, EDGE_TYPE edge_val)`

typedef for the definition of function arguments

## 65 MB\_Shift32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

### Defines

- #define **EDGE\_TYPE\_PIX32**

### Functions

- **MB\_errcode MB\_Shift32** (**MB\_Image** \*src, **MB\_Image** \*dest, **Uint32** dirnum, **Uint32** count, **Uint32** long\_filler\_pix, enum **MB\_grid\_t** grid)

### 65.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 65.2 Define Documentation

#### 65.2.1 #define EDGE\_TYPE\_PIX32

Data type of the value used to represent the edge

### 65.3 Function Documentation

#### 65.3.1 **MB\_errcode MB\_Shift32** ( **MB\_Image** \* *src*, **MB\_Image** \* *dest*, **Uint32** *dirnum*, **Uint32** *count*, **Uint32** *long\_filler\_pix*, enum **MB\_grid\_t** *grid* )

Shifts the content of a 32-bits image in a given direction with a given amplitude The direction depends on the grid used (see MB\_ngh.h for definitions of directions).

#### Parameters

*src* source image  
*dest* destination image  
*dirnum* the direction index  
*count* the amplitude of the shift  
*long\_filler\_pix* the value used to fill the created space  
*grid* the grid used (either square or hexagonal)

#### Returns

An error code (NO\_ERR if successful)

## 66 MB\_Shift8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

## Defines

- `#define EDGE_TYPE Uint32`

## Functions

- `MB_errcode MB_Shift8 (MB_Image *src, MB_Image *dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid)`

## 66.1 Detailed Description

### Author

Nicolas Beucher

### Date

1-5-2010

## 66.2 Define Documentation

### 66.2.1 `#define EDGE_TYPE Uint32`

Data type of the value used to represent the edge

## 66.3 Function Documentation

### 66.3.1 `MB_errcode MB_Shift8 ( MB_Image * src, MB_Image * dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid )`

Shifts the contents of a 8-bits image in a given direction with a given amplitude The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

### Parameters

- src* source image
- dest* destination image
- dirnum* the direction index
- count* the amplitude of the shift
- long\_filler\_pix* the value used to fill the created space
- grid* the grid used (either square or hexagonal)

### Returns

An error code (`NO_ERR` if successful)

## 67 MB\_Shiftb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

## Defines

- `#define EDGE_TYPE binaryT`

## Functions

- `MB_errcode MB_Shiftb (MB_Image *src, MB_Image *dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid)`

### 67.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-7-2008

### 67.2 Define Documentation

#### 67.2.1 `#define EDGE_TYPE binaryT`

Data type of the value used to represent the edge

### 67.3 Function Documentation

#### 67.3.1 `MB_errcode MB_Shiftb ( MB_Image * src, MB_Image * dest, Uint32 dirnum, Uint32 count, Uint32 long_filler_pix, enum MB_grid_t grid )`

Shifts the source image in the given direction. The direction depends on the grid used (see `MB_ngh.h` for definitions of directions).

#### Parameters

*src* source image

*dest* destination image

*dirnum* the direction (from 0 to 8)

*count* the amplitude of the shift (in pixels)

*long\_filler\_pix* the value used to fill the created space

*grid* the grid used (either square or hexagonal)

#### Returns

An error code (`NO_ERR` if successful)

## 68 MB\_Sub.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Sub (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

## 68.1 Detailed Description

### Author

Nicolas Beucher

### Date

13-6-2007

## 68.2 Function Documentation

### 68.2.1 `MB_errcode MB_Sub ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Subtracts the values of pixels of the second image to the values of the pixels in the first image

#### Parameters

*src1* image 1

*src2* image 2

*dest* image resulting of the subtraction

#### Returns

An error code (NO\_ERR if successful)

## 69 MB\_Sup.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Sup (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

## 69.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-25-2007

## 69.2 Function Documentation

### 69.2.1 `MB_errcode MB_Sup ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Determines the superior value between the pixels of two images. The result is put in the corresponding pixel position in the destination image.

#### Parameters

*src1* image 1

*src2* image 2

*dest* destination image

## Returns

An error code (NO\_ERR if successful)

## 70 MB\_SupFarNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

## Defines

- #define EDGE\_TYPE\_PIX32

## Functions

- MB\_errcode MB\_SupFarNb32 (MB\_Image \*src, MB\_Image \*srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge)

### 70.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

29-6-2010

### 70.2 Define Documentation

#### 70.2.1 #define EDGE\_TYPE\_PIX32

Data type of the value used to represent the edge

### 70.3 Function Documentation

#### 70.3.1 MB\_errcode MB\_SupFarNb32 ( MB\_Image \* src, MB\_Image \* srcdest, Uint32 nbrnum, Uint32 count, enum MB\_grid\_t grid, enum MB\_edgemode\_t edge )

Looks for the maximum between two 32-bits image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

#### Parameters

- src* source image in which the neighbor are taken
- srcdest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* the amplitude of the shift (in pixels)
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (NO\_ERR if successful)

## 71 MB\_SupFarNb8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

### Defines

- `#define EDGE_TYPE Uint32`

### Functions

- `MB_errcode MB_SupFarNb8 (MB_Image *src, MB_Image *srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 71.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

25-6-2010

### 71.2 Define Documentation

#### 71.2.1 `#define EDGE_TYPE Uint32`

Data type of the value used to represent the edge

### 71.3 Function Documentation

#### 71.3.1 `MB_errcode MB_SupFarNb8 ( MB_Image * src, MB_Image * srcdest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two grey scale image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see `MB_ngh.h`).

#### Parameters

*src* source image in which the neighbor are taken  
*srcdest* source of the central pixel and destination image  
*nbrnum* the neighbor index  
*count* the amplitude of the shift (in pixels)  
*grid* the grid used (either square or hexagonal)  
*edge* the kind of edge to use (behavior for pixels near edge depends on it)

#### Returns

An error code (`NO_ERR` if successful)

## 72 MB\_SupFarNbb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_ShftDirection.c"
```

## Defines

- #define **EDGE\_TYPE** binaryT

## Functions

- **MB\_errcode** **MB\_SupFarNbb** (**MB\_Image** \*src, **MB\_Image** \*srcdest, **Uint32** nbrnum, **Uint32** count, enum **MB\_grid\_t** grid, enum **MB\_edgemode\_t** edge)

## 72.1 Detailed Description

### Author

Nicolas Beucher

### Date

24-6-2010

## 72.2 Define Documentation

### 72.2.1 #define **EDGE\_TYPE** binaryT

Data type of the value used to represent the edge

## 72.3 Function Documentation

### 72.3.1 **MB\_errcode** **MB\_SupFarNbb** ( **MB\_Image** \* *src*, **MB\_Image** \* *srcdest*, **Uint32** *nbrnum*, **Uint32** *count*, enum **MB\_grid\_t** *grid*, enum **MB\_edgemode\_t** *edge* )

Looks for the maximum between two binary image pixels (a central pixel and its far neighbor in the other image) The neighbor depends on the grid used (see **MB\_ngh.h**).

### Parameters

- src* source image in which the neighbor are taken
- srcdest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* the amplitude of the shift (in pixels)
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (**NO\_ERR** if successful)

## 73 MB\_SupMask.c File Reference

```
#include "mambaApi_loc.h"
```

## Functions

- **MB\_errcode** **MB\_SupMask** (**MB\_Image** \*src1, **MB\_Image** \*src2, **MB\_Image** \*dest, **Uint32** strict)

## 73.1 Detailed Description

### Author

Nicolas Beucher

### Date

10-21-2009

## 73.2 Function Documentation

### 73.2.1 `MB_errcode MB_SupMask ( MB_Image * src1, MB_Image * src2, MB_Image * dest, Uint32 strict )`

Computes a binary image where pixels are set to 1 when the pixels of image 1 have greater values than pixels of image 2 otherwise 0

#### Parameters

- src1* source image 1
- src2* source image 2
- dest* destination image
- strict* flag indicating if the comparison is strict or large

#### Returns

An error code (NO\_ERR if successful)

## 74 MB\_SupNb32.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

### Defines

- `#define EDGE_TYPE_PIX32`

### Functions

- `MB_errcode MB_SupNb32 (MB_Image *src, MB_Image *srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

## 74.1 Detailed Description

### Author

Nicolas Beucher

### Date

6-18-2008

## 74.2 Define Documentation

### 74.2.1 `#define EDGE_TYPE_PIX32`

Data type of the value used to represent the edge

## 74.3 Function Documentation

**74.3.1** `MB_errcode MB_SupNb32 ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two 32-bits image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

### Parameters

- src* source image in which the neighbor are taken
- srctest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* if src and srctest are the same, the operation is repeated count times
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

## 75 MB\_SupNb8.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

### Defines

- `#define EDGE_TYPE Uint32`

### Functions

- `MB_errcode MB_SupNb8 (MB_Image *src, MB_Image *srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 75.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 75.2 Define Documentation

**75.2.1** `#define EDGE_TYPE Uint32`

Data type of the value used to represent the edge

## 75.3 Function Documentation

**75.3.1** `MB_errcode MB_SupNb8 ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two greys cale image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

### Parameters

- src* source image in which the neighbor are taken
- srctest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* if src and srctest are the same, the operation is repeated count times
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

## 76 MB\_SupNbb.c File Reference

```
#include "mambaApi_loc.h"
#include "MB_Neighbor.c"
```

### Defines

- `#define EDGE_TYPE binaryT`

### Functions

- `MB_errcode MB_SupNbb (MB_Image *src, MB_Image *srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge)`

### 76.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-18-2008

### 76.2 Define Documentation

**76.2.1** `#define EDGE_TYPE binaryT`

Data type of the value used to represent the edge

## 76.3 Function Documentation

**76.3.1** `MB_errcode MB_SupNbb ( MB_Image * src, MB_Image * srctest, Uint32 nbrnum, Uint32 count, enum MB_grid_t grid, enum MB_edgemode_t edge )`

Looks for the maximum between two binary image pixels (a central pixel and its neighbor in the other image) The neighbor depends on the grid used (see MB\_ngh.h).

### Parameters

- src* source image in which the neighbor are taken
- srctest* source of the central pixel and destination image
- nbrnum* the neighbor index
- count* if src and srctest are the same, the operation is repeated count times
- grid* the grid used (either square or hexagonal)
- edge* the kind of edge to use (behavior for pixels near edge depends on it)

### Returns

An error code (NO\_ERR if successful)

## 77 MB\_Thresh.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Thresh (MB_Image *src, MB_Image *dest, Uint32 low, Uint32 high)`

### 77.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

6-7-2008

### 77.2 Function Documentation

**77.2.1** `MB_errcode MB_Thresh ( MB_Image * src, MB_Image * dest, Uint32 low, Uint32 high )`

Fills a binary image according to the following rules : if pixel value lower than low or higher than high the binary pixel is set to 0, in other cases the pixel is set to 1.

### Parameters

- src* source image
- dest* destination image
- low* low value for threshold
- high* high value for treshold

### Returns

An error code (NO\_ERR if successful)

## 78 MB\_Utils.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- void \* **MB\_malloc** (int size)
- void \* **MB\_aligned\_malloc** (int size, int alignment)
- void **MB\_free** (void \*ptr)
- void **MB\_aligned\_free** (void \*ptr)
- void \* **MB\_memset** (void \*s, int c, int size)
- void \* **MB\_memcpy** (void \*dest, const void \*src, int size)

### 78.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-24-2007

### 78.2 Function Documentation

#### 78.2.1 void MB\_aligned\_free ( void \* *ptr* )

Frees aligned memory.

#### Parameters

*ptr* pointer to the memory space to free

#### 78.2.2 void\* MB\_aligned\_malloc ( int *size*, int *alignment* )

Allocates aligned memory (faster and safer access for SSE2 instructions for example).

#### Parameters

*size* size in bytes of the allocated memory

*alignment* value to which the memory space is aligned

#### Returns

a pointer to the memory space or NULL if unsuccessful

#### 78.2.3 void MB\_free ( void \* *ptr* )

Frees memory.

#### Parameters

*ptr* pointer to the memory space to free

#### 78.2.4 void\* MB\_malloc ( int *size* )

Allocates memory.

##### Parameters

*size* size in byte of the allocated memory

##### Returns

a pointer to the memory space or NULL if unsuccessful

#### 78.2.5 void\* MB\_memcpy ( void \* *dest*, const void \* *src*, int *size* )

Copies a memory space to another. This copy should be protected for safe aliased memory copy.

##### Parameters

*dest* pointer to the destination memory space

*src* pointer to the source memory space

*size* the size of the copy

##### Returns

a pointer to the destination memory space

#### 78.2.6 void\* MB\_memset ( void \* *s*, int *c*, int *size* )

Sets a memory space to a specific value.

##### Parameters

*s* pointer to the memory space to set

*c* value to use to set

*size* the memory space size

##### Returns

a pointer to the memory space set

## 79 MB\_Volume.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- MB\_errcode MB\_Volume (MB\_Image \*src, Uint64 \*pVolume)

### Variables

- const Uint64 MB\_VolumePerByte [256]

## 79.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-25-2007

## 79.2 Function Documentation

### 79.2.1 MB\_errcode MB\_Volume ( MB\_Image \* *src*, Uint64 \* *pVolume* )

Computes the volume of an image. The volume is the sum of the pixel values (i.e. integration of the image)

#### Parameters

*src* source image

*pVolume* pointer to the volume variable

#### Returns

An error code (NO\_ERR if successful)

## 79.3 Variable Documentation

### 79.3.1 const Uint64 MB\_VolumePerByte[256]

#### Initial value:

```
{
    00,01,01,02,01,02,02,03,01,02,02,03,02,03,03,04,
    01,02,02,03,02,03,03,04,02,03,03,04,03,04,04,05,
    01,02,02,03,02,03,03,04,02,03,03,04,03,04,04,05,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    01,02,02,03,02,03,03,04,02,03,03,04,03,04,04,05,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    03,04,04,05,04,05,05,06,04,05,05,06,05,06,06,07,
    01,02,02,03,02,03,03,04,02,03,03,04,03,04,04,05,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    03,04,04,05,04,05,05,06,04,05,05,06,05,06,06,07,
    02,03,03,04,03,04,04,05,03,04,04,05,04,05,05,06,
    03,04,04,05,04,05,05,06,04,05,05,06,05,06,06,07,
    03,04,04,05,04,05,05,06,04,05,05,06,05,06,06,07,
    04,05,05,06,05,06,06,07,05,06,06,07,06,07,07,8
}
```

Volume arrays

## 80 MB\_Watershed.c File Reference

```
#include "mambaApi_loc.h"
```

### Data Structures

- struct MB\_Watershed\_Ctx

## Typedefs

- typedef `MB_Token( TSWITCHEP )(void *ctx, int x, int y)`

## Functions

- `MB_errcode MB_Watershed (MB_Image *src, MB_Image *marker, Uint32 max_level, enum MB_grid_t grid)`

## Variables

- const int `sqNbDir` [9][2]
- const int `hxNbDir` [2][7][2]

## 80.1 Detailed Description

### Author

Nicolas Beucher

### Date

11-10-2009

## 80.2 Typedef Documentation

### 80.2.1 typedef MB\_Token( TSWITCHEP)(void \*ctx, int x, int y)

typedef for the definition of neighbor function arguments

## 80.3 Function Documentation

### 80.3.1 MB\_errcode MB\_Watershed ( MB\_Image \* src, MB\_Image \* marker, Uint32 max\_level, enum MB\_grid\_t grid )

Performs a watershed segmentation of the image using the marker image as a starting point for the flooding. The function builds the actual watershed line (idempotent) plus catchment basins (not idempotent). The result is put into the 32-bits marker image.

The segmentation is coded as follows into the 32-bits values. | 0 | 1 | 2 | 3 | |-----|-----|-----|-----| | Segment label | isLine | Each byte can be accessed using the function `MB_CopyBytePlane`. `isLine` is a value indicating if the pixel belongs to the watershed (255 if this is the case, undefined otherwise).

### Parameters

- *src* the greyscale image to segment
- *marker* the marker image in which the result of segmentation will be put
- *max\_level* the maximum level reach by the water.
- *grid* the grid used (either square or hexagonal)

### Returns

An error code (`NO_ERR` if successful)

## 80.4 Variable Documentation

### 80.4.1 `const int hxNbDir[2][7][2]`

**Initial value:**

```
{
  {{0,0},{0,-1},{1,0},{0,1},{-1,1},{-1,0},{-1,-1}},
  {{0,0},{1,-1},{1,0},{1,1},{0,1},{-1,0},{0,-1}}
}
```

Table giving the offset for the neighbor in hexagonal grid (x and y)

### 80.4.2 `const int sqNbDir[9][2]`

**Initial value:**

```
{
  {0,0},{0,-1},{1,-1},{1,0},{1,1},{0,1},{-1,1},{-1,0},{-1,-1}
}
```

Table giving the offset for the neighbor in square grid (x and y)

## 81 MB\_Xor.c File Reference

```
#include "mambaApi_loc.h"
```

### Functions

- `MB_errcode MB_Xor (MB_Image *src1, MB_Image *src2, MB_Image *dest)`

### 81.1 Detailed Description

#### Author

Nicolas Beucher

#### Date

11-24-2007

### 81.2 Function Documentation

**81.2.1** `MB_errcode MB_Xor ( MB_Image * src1, MB_Image * src2, MB_Image * dest )`

Applies a XOR on the pixels of two images. All the images must have the same depth for correct work.

#### Parameters

*src1* image 1  
*src2* image 2  
*dest* destination image

#### Returns

An error code (NO\_ERR if successful)

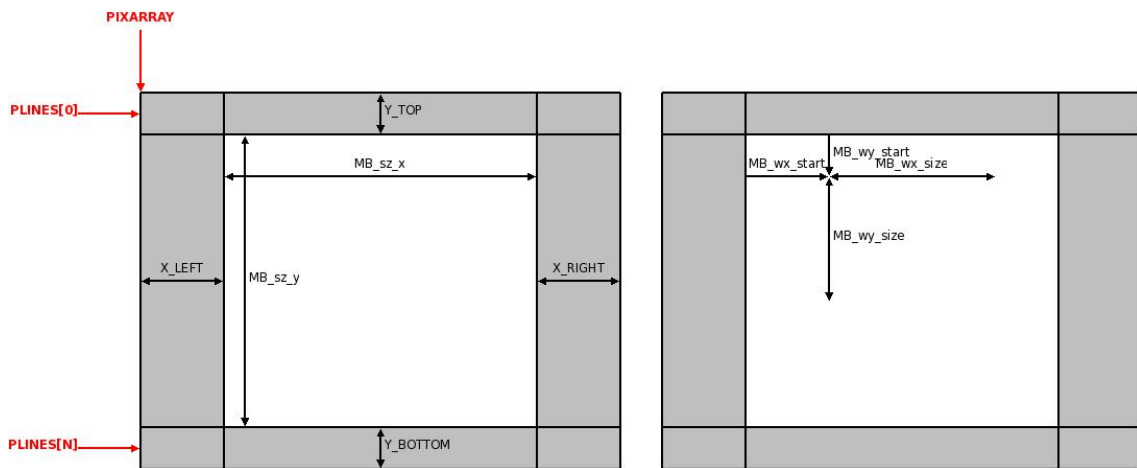


Figure 1: Image structure and variables