

ECVL

Generated by Doxygen 1.8.15



<b>1 Documentation</b>	<b>1</b>
<b>2 Bug List</b>	<b>3</b>
<b>3 Namespace Index</b>	<b>5</b>
3.1 Namespace List . . . . .	5
<b>4 Hierarchical Index</b>	<b>7</b>
4.1 Class Hierarchy . . . . .	7
<b>5 Class Index</b>	<b>9</b>
5.1 Class List . . . . .	9
<b>6 File Index</b>	<b>11</b>
6.1 File List . . . . .	11
<b>7 Namespace Documentation</b>	<b>13</b>
7.1 ecvl Namespace Reference . . . . .	13
7.1.1 Enumeration Type Documentation . . . . .	16
7.1.1.1 ColorType . . . . .	16
7.1.1.2 DataType . . . . .	17
7.1.1.3 InterpolationType . . . . .	17
7.1.1.4 ThresholdingType . . . . .	18
7.1.2 Function Documentation . . . . .	18
7.1.2.1 Add() [1/4] . . . . .	18
7.1.2.2 Add() [2/4] . . . . .	18
7.1.2.3 Add() [3/4] . . . . .	19
7.1.2.4 Add() [4/4] . . . . .	19
7.1.2.5 ChangeColorSpace() . . . . .	20
7.1.2.6 CopyImage() . . . . .	20
7.1.2.7 DataTypeArray() . . . . .	21
7.1.2.8 DataTypeSignedArray() . . . . .	21
7.1.2.9 DataTypeSignedSize() . . . . .	21
7.1.2.10 DataTypeSize() [1/2] . . . . .	22
7.1.2.11 DataTypeSize() [2/2] . . . . .	22
7.1.2.12 Div() [1/2] . . . . .	22
7.1.2.13 Div() [2/2] . . . . .	23
7.1.2.14 Flip2D() . . . . .	23
7.1.2.15 ImageToMat() . . . . .	23
7.1.2.16 ImRead() [1/2] . . . . .	24
7.1.2.17 ImRead() [2/2] . . . . .	24
7.1.2.18 ImShow() . . . . .	25
7.1.2.19 ImWrite() [1/2] . . . . .	25
7.1.2.20 ImWrite() [2/2] . . . . .	26

7.1.2.21 MatToImage()	26
7.1.2.22 Mirror2D()	26
7.1.2.23 Mul() [1/4]	27
7.1.2.24 Mul() [2/4]	27
7.1.2.25 Mul() [3/4]	27
7.1.2.26 Mul() [4/4]	28
7.1.2.27 Neg()	28
7.1.2.28 OtsuThreshold()	29
7.1.2.29 RearrangeChannels()	29
7.1.2.30 ResizeDim()	30
7.1.2.31 ResizeScale()	30
7.1.2.32 Rotate2D()	31
7.1.2.33 RotateFullImage2D()	31
7.1.2.34 saturate_cast() [1/2]	32
7.1.2.35 saturate_cast() [2/2]	32
7.1.2.36 Sub() [1/4]	33
7.1.2.37 Sub() [2/4]	33
7.1.2.38 Sub() [3/4]	33
7.1.2.39 Sub() [4/4]	34
7.1.2.40 Threshold()	34
7.1.2.41 wx_from_mat()	35
7.2 filesystem Namespace Reference	35
7.2.1 Function Documentation	35
7.2.1.1 copy() [1/3]	36
7.2.1.2 copy() [2/3]	36
7.2.1.3 copy() [3/3]	36
7.2.1.4 create_directories() [1/3]	36
7.2.1.5 create_directories() [2/3]	36
7.2.1.6 create_directories() [3/3]	37
7.2.1.7 exists() [1/3]	37
7.2.1.8 exists() [2/3]	37
7.2.1.9 exists() [3/3]	37
7.2.1.10 operator/()	37
<b>8 Class Documentation</b>	<b>39</b>
8.1 ecvl::ConstContiguousIterator< T > Struct Template Reference	39
8.1.1 Detailed Description	39
8.1.2 Constructor & Destructor Documentation	39
8.1.2.1 ConstContiguousIterator()	40
8.1.3 Member Function Documentation	40
8.1.3.1 operator*()	40
8.1.3.2 operator!=(=)	40

8.1.3.3 operator++()	40
8.1.3.4 operator->()	40
8.1.3.5 operator==()	41
8.1.4 Member Data Documentation	41
8.1.4.1 img_	41
8.1.4.2 ptr_	41
8.2 ecvl::ConstContiguousView< DT > Class Template Reference	41
8.2.1 Detailed Description	42
8.2.2 Member Typedef Documentation	42
8.2.2.1 basetype	42
8.2.3 Constructor & Destructor Documentation	42
8.2.3.1 ConstContiguousView()	42
8.2.4 Member Function Documentation	43
8.2.4.1 Begin()	43
8.2.4.2 End()	43
8.2.4.3 operator()()	43
8.3 ecvl::ConstIterator< T > Struct Template Reference	43
8.3.1 Detailed Description	44
8.3.2 Member Typedef Documentation	44
8.3.2.1 IncrementMemFn	44
8.3.3 Constructor & Destructor Documentation	44
8.3.3.1 ConstIterator()	44
8.3.4 Member Function Documentation	45
8.3.4.1 operator *()	45
8.3.4.2 operator"!=(	45
8.3.4.3 operator++()	45
8.3.4.4 operator->()	45
8.3.4.5 operator==(	45
8.3.5 Member Data Documentation	46
8.3.5.1 img_	46
8.3.5.2 incrementor	46
8.3.5.3 pos_	46
8.3.5.4 ptr_	46
8.4 ecvl::ConstView< DT > Class Template Reference	47
8.4.1 Detailed Description	47
8.4.2 Member Typedef Documentation	47
8.4.2.1 basetype	47
8.4.3 Constructor & Destructor Documentation	47
8.4.3.1 ConstView()	48
8.4.4 Member Function Documentation	48
8.4.4.1 Begin()	48
8.4.4.2 End()	48

8.4.4.3 operator>()	48
8.5 ecvl::ContiguousIterator< T > Struct Template Reference	48
8.5.1 Detailed Description	49
8.5.2 Constructor & Destructor Documentation	49
8.5.2.1 ContiguousIterator()	49
8.5.3 Member Function Documentation	49
8.5.3.1 operator *()	49
8.5.3.2 operator !=()	50
8.5.3.3 operator ++()	50
8.5.3.4 operator ->()	50
8.5.3.5 operator ==()	50
8.5.4 Member Data Documentation	50
8.5.4.1 img_	50
8.5.4.2 ptr_	51
8.6 ecvl::ContiguousView< DT > Class Template Reference	51
8.6.1 Detailed Description	51
8.6.2 Member Typedef Documentation	52
8.6.2.1 basetype	52
8.6.3 Constructor & Destructor Documentation	52
8.6.3.1 ContiguousView()	52
8.6.4 Member Function Documentation	52
8.6.4.1 Begin()	52
8.6.4.2 End()	52
8.6.4.3 operator>()	53
8.7 ecvl::ContiguousViewXYC< DT > Class Template Reference	53
8.7.1 Detailed Description	53
8.7.2 Member Typedef Documentation	54
8.7.2.1 basetype	54
8.7.3 Constructor & Destructor Documentation	54
8.7.3.1 ContiguousViewXYC()	54
8.7.4 Member Function Documentation	54
8.7.4.1 Begin()	54
8.7.4.2 channels()	54
8.7.4.3 End()	55
8.7.4.4 height()	55
8.7.4.5 operator>()	55
8.7.4.6 width()	55
8.8 DefaultMemoryManager Class Reference	55
8.8.1 Detailed Description	56
8.8.2 Member Function Documentation	56
8.8.2.1 Allocate()	56
8.8.2.2 AllocateAndCopy()	56

8.8.2.3 Deallocate()	56
8.8.2.4 GetInstance()	57
8.9 ecvl::DivImpl< ST1, ST2, ET > Struct Template Reference	57
8.9.1 Detailed Description	57
8.9.2 Member Function Documentation	57
8.9.2.1 _()	57
8.10 ecvl::DivImpl< Image, Image, ET > Struct Template Reference	58
8.10.1 Detailed Description	58
8.10.2 Member Function Documentation	58
8.10.2.1 _()	58
8.11 ecvl::DivImpl< Image, T, ET > Struct Template Reference	58
8.11.1 Detailed Description	59
8.11.2 Member Function Documentation	59
8.11.2.1 _()	59
8.12 ecvl::DivImpl< T, Image, ET > Struct Template Reference	59
8.12.1 Detailed Description	59
8.12.2 Member Function Documentation	59
8.12.2.1 _()	60
8.13 ecvl::Image Class Reference	60
8.13.1 Detailed Description	62
8.13.2 Constructor & Destructor Documentation	62
8.13.2.1 Image() [1/4]	62
8.13.2.2 Image() [2/4]	62
8.13.2.3 Image() [3/4]	62
8.13.2.4 Image() [4/4]	63
8.13.2.5 ~Image()	63
8.13.3 Member Function Documentation	63
8.13.3.1 Begin() [1/2]	63
8.13.3.2 Begin() [2/2]	63
8.13.3.3 ContiguousBegin() [1/2]	64
8.13.3.4 ContiguousBegin() [2/2]	64
8.13.3.5 ContiguousEnd() [1/2]	64
8.13.3.6 ContiguousEnd() [2/2]	64
8.13.3.7 Create()	65
8.13.3.8 End() [1/2]	65
8.13.3.9 End() [2/2]	65
8.13.3.10 IsEmpty()	66
8.13.3.11 IsOwner()	66
8.13.3.12 operator=()	66
8.13.3.13 Ptr() [1/2]	66
8.13.3.14 Ptr() [2/2]	66
8.13.4 Friends And Related Function Documentation	67

8.13.4.1 swap . . . . .	67
8.13.5 Member Data Documentation . . . . .	67
8.13.5.1 channels_ . . . . .	67
8.13.5.2 colortype_ . . . . .	67
8.13.5.3 contiguous_ . . . . .	68
8.13.5.4 data_ . . . . .	68
8.13.5.5 datasize_ . . . . .	68
8.13.5.6 dims_ . . . . .	68
8.13.5.7 elemsize_ . . . . .	68
8.13.5.8 elemtype_ . . . . .	69
8.13.5.9 mem_ . . . . .	69
8.13.5.10 meta_ . . . . .	69
8.13.5.11 strides_ . . . . .	69
8.14 ecvl::Table1D< _StructFun, Args >::integer< i > Struct Template Reference . . . . .	70
8.14.1 Detailed Description . . . . .	70
8.15 ecvl::SignedTable1D< _StructFun, Args >::integer< i > Struct Template Reference . . . . .	70
8.15.1 Detailed Description . . . . .	70
8.16 ecvl::Table2D< _StructFun, Args >::integer< i > Struct Template Reference . . . . .	70
8.16.1 Detailed Description . . . . .	70
8.17 ecvl::Iterator< T > Struct Template Reference . . . . .	71
8.17.1 Detailed Description . . . . .	71
8.17.2 Member Typedef Documentation . . . . .	71
8.17.2.1 IncrementMemFn . . . . .	71
8.17.3 Constructor & Destructor Documentation . . . . .	71
8.17.3.1 Iterator() . . . . .	72
8.17.4 Member Function Documentation . . . . .	72
8.17.4.1 operator*() . . . . .	72
8.17.4.2 operator"!=( ) . . . . .	72
8.17.4.3 operator++() . . . . .	72
8.17.4.4 operator->() . . . . .	72
8.17.4.5 operator==( ) . . . . .	73
8.17.5 Member Data Documentation . . . . .	73
8.17.5.1 img_ . . . . .	73
8.17.5.2 incrementor . . . . .	73
8.17.5.3 pos_ . . . . .	73
8.17.5.4 ptr_ . . . . .	73
8.18 MemoryManager Class Reference . . . . .	74
8.18.1 Detailed Description . . . . .	74
8.18.2 Constructor & Destructor Documentation . . . . .	74
8.18.2.1 ~MemoryManager() . . . . .	74
8.18.3 Member Function Documentation . . . . .	74
8.18.3.1 Allocate() . . . . .	74



8.18.3.2 AllocateAndCopy()	75
8.18.3.3 Deallocate()	75
8.19 ecvl::MetaData Class Reference	75
8.19.1 Detailed Description	75
8.19.2 Constructor & Destructor Documentation	75
8.19.2.1 ~MetaData()	75
8.19.3 Member Function Documentation	76
8.19.3.1 Query()	76
8.20 filesystem::path Class Reference	76
8.20.1 Detailed Description	76
8.20.2 Constructor & Destructor Documentation	76
8.20.2.1 path() [1/2]	76
8.20.2.2 path() [2/2]	77
8.20.3 Member Function Documentation	77
8.20.3.1 operator/=( )	77
8.20.3.2 operator=( ) [1/2]	77
8.20.3.3 operator=( ) [2/2]	77
8.20.3.4 parent_path()	77
8.20.3.5 stem()	78
8.20.3.6 string()	78
8.21 ShallowMemoryManager Class Reference	78
8.21.1 Detailed Description	78
8.21.2 Member Function Documentation	79
8.21.2.1 Allocate()	79
8.21.2.2 AllocateAndCopy()	79
8.21.2.3 Deallocate()	79
8.21.2.4 GetInstance()	79
8.22 ecvl::ShowApp Class Reference	80
8.22.1 Detailed Description	80
8.22.2 Constructor & Destructor Documentation	80
8.22.2.1 ShowApp()	80
8.22.3 Member Function Documentation	80
8.22.3.1 OnInit()	81
8.23 ecvl::SignedTable1D<_StructFun, Args> Struct Template Reference	81
8.23.1 Detailed Description	81
8.23.2 Member Typedef Documentation	82
8.23.2.1 fun_type	82
8.23.3 Constructor & Destructor Documentation	82
8.23.3.1 SignedTable1D()	82
8.23.4 Member Function Documentation	82
8.23.4.1 FillData() [1/2]	82
8.23.4.2 FillData() [2/2]	82

8.23.4.3 operator>()	83
8.23.5 Member Data Documentation	83
8.23.5.1 data	83
8.24 ecvl::StructAdd< a, b > Struct Template Reference	83
8.24.1 Detailed Description	83
8.24.2 Member Function Documentation	83
8.24.2.1 _()	84
8.25 ecvl::StructCopyImage< SDT, DDT > Struct Template Reference	84
8.25.1 Detailed Description	84
8.25.2 Member Function Documentation	84
8.25.2.1 _()	84
8.26 ecvl::StructDiv< a, b, ET > Struct Template Reference	85
8.26.1 Detailed Description	85
8.26.2 Member Function Documentation	85
8.26.2.1 _()	85
8.27 ecvl::StructMul< a, b > Struct Template Reference	85
8.27.1 Detailed Description	85
8.27.2 Member Function Documentation	86
8.27.2.1 _()	86
8.28 ecvl::StructScalarAdd< DT, T > Struct Template Reference	86
8.28.1 Detailed Description	86
8.28.2 Member Function Documentation	86
8.28.2.1 _()	86
8.29 ecvl::StructScalarDiv< DT, T > Struct Template Reference	87
8.29.1 Detailed Description	87
8.29.2 Member Function Documentation	87
8.29.2.1 _()	87
8.30 ecvl::StructScalarDivInv< DT, T, ET > Struct Template Reference	87
8.30.1 Detailed Description	88
8.30.2 Member Function Documentation	88
8.30.2.1 _()	88
8.31 ecvl::StructScalarMul< DT, T > Struct Template Reference	88
8.31.1 Detailed Description	89
8.31.2 Member Function Documentation	89
8.31.2.1 _()	89
8.32 ecvl::StructScalarNeg< DT > Struct Template Reference	89
8.32.1 Detailed Description	89
8.32.2 Member Function Documentation	89
8.32.2.1 _()	90
8.33 ecvl::StructScalarSub< DT, T > Struct Template Reference	90
8.33.1 Detailed Description	90
8.33.2 Member Function Documentation	90

8.33.2.1 _()	90
8.34 ecvl::StructScalarSubInv< DT, T > Struct Template Reference	91
8.34.1 Detailed Description	91
8.34.2 Member Function Documentation	91
8.34.2.1 _()	91
8.35 ecvl::StructSub< a, b > Struct Template Reference	91
8.35.1 Detailed Description	91
8.35.2 Member Function Documentation	92
8.35.2.1 _()	92
8.36 ecvl::Table1D< _StructFun, Args > Struct Template Reference	92
8.36.1 Detailed Description	92
8.36.2 Member Typedef Documentation	93
8.36.2.1 fun_type	93
8.36.3 Constructor & Destructor Documentation	93
8.36.3.1 Table1D()	93
8.36.4 Member Function Documentation	93
8.36.4.1 FillData() [1/2]	93
8.36.4.2 FillData() [2/2]	93
8.36.4.3 operator>()	94
8.36.5 Member Data Documentation	94
8.36.5.1 data	94
8.37 ecvl::Table2D< _StructFun, Args > Struct Template Reference	94
8.37.1 Detailed Description	95
8.37.2 Member Typedef Documentation	95
8.37.2.1 fun_type	95
8.37.3 Constructor & Destructor Documentation	95
8.37.3.1 Table2D()	95
8.37.4 Member Function Documentation	95
8.37.4.1 FillData() [1/2]	95
8.37.4.2 FillData() [2/2]	96
8.37.4.3 operator>()	96
8.37.5 Member Data Documentation	96
8.37.5.1 data	96
8.38 ecvl::TypeInfo< DataType > Struct Template Reference	96
8.38.1 Detailed Description	96
8.38.2 Member Typedef Documentation	97
8.38.2.1 basetype	97
8.39 ecvl::TypeInfo< ecvl::DataType::float32 > Struct Template Reference	97
8.39.1 Detailed Description	97
8.39.2 Member Typedef Documentation	97
8.39.2.1 basetype	97
8.40 ecvl::TypeInfo< ecvl::DataType::float64 > Struct Template Reference	98

8.40.1 Detailed Description . . . . .	98
8.40.2 Member Typedef Documentation . . . . .	98
8.40.2.1 basetype . . . . .	98
8.41 <code>ecvl::TypeInfo&lt; ecvl::DataType::int16 &gt;</code> Struct Template Reference . . . . .	98
8.41.1 Detailed Description . . . . .	98
8.41.2 Member Typedef Documentation . . . . .	99
8.41.2.1 basetype . . . . .	99
8.42 <code>ecvl::TypeInfo&lt; ecvl::DataType::int32 &gt;</code> Struct Template Reference . . . . .	99
8.42.1 Detailed Description . . . . .	99
8.42.2 Member Typedef Documentation . . . . .	99
8.42.2.1 basetype . . . . .	99
8.43 <code>ecvl::TypeInfo&lt; ecvl::DataType::int64 &gt;</code> Struct Template Reference . . . . .	100
8.43.1 Detailed Description . . . . .	100
8.43.2 Member Typedef Documentation . . . . .	100
8.43.2.1 basetype . . . . .	100
8.44 <code>ecvl::TypeInfo&lt; ecvl::DataType::int8 &gt;</code> Struct Template Reference . . . . .	100
8.44.1 Detailed Description . . . . .	100
8.44.2 Member Typedef Documentation . . . . .	101
8.44.2.1 basetype . . . . .	101
8.45 <code>ecvl::TypeInfo&lt; ecvl::DataType::none &gt;</code> Struct Template Reference . . . . .	101
8.45.1 Detailed Description . . . . .	101
8.45.2 Member Typedef Documentation . . . . .	101
8.45.2.1 basetype . . . . .	101
8.46 <code>ecvl::TypeInfo&lt; ecvl::DataType::uint16 &gt;</code> Struct Template Reference . . . . .	102
8.46.1 Detailed Description . . . . .	102
8.46.2 Member Typedef Documentation . . . . .	102
8.46.2.1 basetype . . . . .	102
8.47 <code>ecvl::TypeInfo&lt; ecvl::DataType::uint32 &gt;</code> Struct Template Reference . . . . .	102
8.47.1 Detailed Description . . . . .	102
8.47.2 Member Typedef Documentation . . . . .	103
8.47.2.1 basetype . . . . .	103
8.48 <code>ecvl::TypeInfo&lt; ecvl::DataType::uint64 &gt;</code> Struct Template Reference . . . . .	103
8.48.1 Detailed Description . . . . .	103
8.48.2 Member Typedef Documentation . . . . .	103
8.48.2.1 basetype . . . . .	103
8.49 <code>ecvl::TypeInfo&lt; ecvl::DataType::uint8 &gt;</code> Struct Template Reference . . . . .	104
8.49.1 Detailed Description . . . . .	104
8.49.2 Member Typedef Documentation . . . . .	104
8.49.2.1 basetype . . . . .	104
8.50 <code>ecvl::View&lt; DT &gt;</code> Class Template Reference . . . . .	104
8.50.1 Detailed Description . . . . .	105
8.50.2 Member Typedef Documentation . . . . .	105

8.50.2.1 basetype . . . . .	105
8.50.3 Constructor & Destructor Documentation . . . . .	105
8.50.3.1 View() [1/2] . . . . .	105
8.50.3.2 View() [2/2] . . . . .	106
8.50.4 Member Function Documentation . . . . .	106
8.50.4.1 Begin() . . . . .	106
8.50.4.2 End() . . . . .	106
8.50.4.3 operator()() . . . . .	106
8.51 ecvl::wxImagePanel Class Reference . . . . .	107
8.51.1 Detailed Description . . . . .	107
8.51.2 Constructor & Destructor Documentation . . . . .	107
8.51.2.1 wxImagePanel() . . . . .	107
8.51.3 Member Function Documentation . . . . .	107
8.51.3.1 SetImage() . . . . .	107
<b>9 File Documentation</b>	<b>109</b>
9.1 arithmetic.cpp File Reference . . . . .	109
9.1.1 Macro Definition Documentation . . . . .	110
9.1.1.1 STANDARD_INPLACE_OPERATION . . . . .	110
9.1.1.2 STANDARD_NON_INPLACE_OPERATION . . . . .	110
9.2 arithmetic.h File Reference . . . . .	110
9.3 core.cpp File Reference . . . . .	112
9.4 core.h File Reference . . . . .	112
9.5 datatype.cpp File Reference . . . . .	112
9.5.1 Macro Definition Documentation . . . . .	113
9.5.1.1 ECVL_TUPLE . . . . .	113
9.6 datatype.h File Reference . . . . .	113
9.6.1 Macro Definition Documentation . . . . .	114
9.6.1.1 ECVL_TUPLE [1/4] . . . . .	114
9.6.1.2 ECVL_TUPLE [2/4] . . . . .	114
9.6.1.3 ECVL_TUPLE [3/4] . . . . .	115
9.6.1.4 ECVL_TUPLE [4/4] . . . . .	115
9.7 datatype_existing_tuples.inc.h File Reference . . . . .	115
9.8 datatype_existing_tuples_signed.inc.h File Reference . . . . .	115
9.9 datatype_existing_tuples_unsigned.inc.h File Reference . . . . .	115
9.10 datatype_matrix.h File Reference . . . . .	115
9.11 datatype_tuples.inc.h File Reference . . . . .	116
9.12 filesystem.cc File Reference . . . . .	116
9.13 filesystem.h File Reference . . . . .	116
9.14 gui.cpp File Reference . . . . .	117
9.15 gui.h File Reference . . . . .	117
9.16 home.h File Reference . . . . .	118

9.17 image.cpp File Reference . . . . .	118
9.18 image.h File Reference . . . . .	118
9.19 imgcodecs.cpp File Reference . . . . .	119
9.20 imgcodecs.h File Reference . . . . .	119
9.21 imgproc.cpp File Reference . . . . .	120
9.22 imgproc.h File Reference . . . . .	121
9.23 iterators.h File Reference . . . . .	122
9.24 iterators_impl.inc.h File Reference . . . . .	122
9.25 memorymanager.cpp File Reference . . . . .	122
9.26 memorymanager.h File Reference . . . . .	122
9.27 standard_errors.h File Reference . . . . .	123
9.27.1 Macro Definition Documentation . . . . .	123
9.27.1.1 ECVL_ERROR_EMPTY_IMAGE . . . . .	123
9.27.1.2 ECVL_ERROR_MSG . . . . .	123
9.27.1.3 ECVL_ERROR_NOT_ALLOWED_ON_NON_OWING_IMAGE . . . . .	123
9.27.1.4 ECVL_ERROR_NOT_ALLOWED_ON_UNSIGNED_IMG . . . . .	124
9.27.1.5 ECVL_ERROR_NOT_IMPLEMENTED . . . . .	124
9.27.1.6 ECVL_ERROR_NOT_REACHABLE_CODE . . . . .	124
9.27.1.7 ECVL_ERROR_UNSUPPORTED_OPENCV_DEPTH . . . . .	124
9.27.1.8 ECVL_ERROR_UNSUPPORTED_OPENCV_DIMS . . . . .	124
9.27.1.9 ECVL_ERROR_WRONG_PARAMS . . . . .	124
9.27.1.10 ECVL_WARNING_MSG . . . . .	125
9.28 support_opencv.cpp File Reference . . . . .	125
9.29 support_opencv.h File Reference . . . . .	125
9.30 test_core.cpp File Reference . . . . .	125
9.30.1 Function Documentation . . . . .	126
9.30.1.1 TEST() [1/2] . . . . .	126
9.30.1.2 TEST() [2/2] . . . . .	126
<b>Index</b>	<b>127</b>

## Chapter 1

# Documentation

ECVL is the European Computer Vision Library, under development within the European project DeepHealth. Here you can find the provisional doxygen documentation. Checkout the GitHub project [here](#).





## Chapter 2

## Bug List

**Member `ecvl::Threshold` (p. 34) (`const Image` (p. 60) `&src`, `Image` (p. 60) `&dst`, `double thresh`, `double maxval`, `ThresholdingType thresh_type=ThresholdingType::BINARY` (p. 18))**

Input and output Images may have different color spaces.



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>ecvl</b>	13
<b>filesystem</b>	35



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ecvl::ConstContiguousIterator< T > . . . . .	39
ecvl::ConstIterator< T > . . . . .	43
ecvl::ContiguousIterator< T > . . . . .	48
ecvl::DivImpl< ST1, ST2, ET > . . . . .	57
ecvl::DivImpl< Image, Image, ET > . . . . .	58
ecvl::DivImpl< Image, T, ET > . . . . .	58
ecvl::DivImpl< T, Image, ET > . . . . .	59
ecvl::Image . . . . .	60
ecvl::ConstContiguousView< DT > . . . . .	41
ecvl::ConstView< DT > . . . . .	47
ecvl::ContiguousView< DT > . . . . .	51
ecvl::ContiguousViewXYC< DT > . . . . .	53
ecvl::View< DT > . . . . .	104
ecvl::Table1D< _StructFun, Args >::integer< i > . . . . .	70
ecvl::SignedTable1D< _StructFun, Args >::integer< i > . . . . .	70
ecvl::Table2D< _StructFun, Args >::integer< i > . . . . .	70
ecvl::Iterator< T > . . . . .	71
MemoryManager . . . . .	74
DefaultMemoryManager . . . . .	55
ShallowMemoryManager . . . . .	78
ecvl::MetaData . . . . .	75
filesystem::path . . . . .	76
ecvl::SignedTable1D< _StructFun, Args > . . . . .	81
ecvl::StructAdd< a, b > . . . . .	83
ecvl::StructCopyImage< SDT, DDT > . . . . .	84
ecvl::StructDiv< a, b, ET > . . . . .	85
ecvl::StructMul< a, b > . . . . .	85
ecvl::StructScalarAdd< DT, T > . . . . .	86
ecvl::StructScalarDiv< DT, T > . . . . .	87
ecvl::StructScalarDivInv< DT, T, ET > . . . . .	87
ecvl::StructScalarMul< DT, T > . . . . .	88
ecvl::StructScalarNeg< DT > . . . . .	89
ecvl::StructScalarSub< DT, T > . . . . .	90
ecvl::StructScalarSubInv< DT, T > . . . . .	91
ecvl::StructSub< a, b > . . . . .	91

ecvl::Table1D< _StructFun, Args > . . . . .	92
ecvl::Table2D< _StructFun, Args > . . . . .	94
ecvl::TypeInfo< DataType > . . . . .	96
ecvl::TypeInfo< ecvl::DataType::float32 > . . . . .	97
ecvl::TypeInfo< ecvl::DataType::float64 > . . . . .	98
ecvl::TypeInfo< ecvl::DataType::int16 > . . . . .	98
ecvl::TypeInfo< ecvl::DataType::int32 > . . . . .	99
ecvl::TypeInfo< ecvl::DataType::int64 > . . . . .	100
ecvl::TypeInfo< ecvl::DataType::int8 > . . . . .	100
ecvl::TypeInfo< ecvl::DataType::none > . . . . .	101
ecvl::TypeInfo< ecvl::DataType::uint16 > . . . . .	102
ecvl::TypeInfo< ecvl::DataType::uint32 > . . . . .	102
ecvl::TypeInfo< ecvl::DataType::uint64 > . . . . .	103
ecvl::TypeInfo< ecvl::DataType::uint8 > . . . . .	104
wxApp	
ecvl::ShowApp . . . . .	80
wxPanel	
ecvl::wxImagePanel . . . . .	107

## Chapter 5

# Class Index

### 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>ecvl::ConstContiguousIterator</b> < T > . . . . .	39
<b>ecvl::ConstContiguousView</b> < DT > . . . . .	41
<b>ecvl::ConstIterator</b> < T > . . . . .	43
<b>ecvl::ConstView</b> < DT > . . . . .	47
<b>ecvl::ContiguousIterator</b> < T > . . . . .	48
<b>ecvl::ContiguousView</b> < DT > . . . . .	51
<b>ecvl::ContiguousViewXYC</b> < DT > . . . . .	53
<b>DefaultMemoryManager</b> . . . . .	55
<b>ecvl::DivImpl</b> < ST1, ST2, ET > . . . . .	57
<b>ecvl::DivImpl</b> < Image, Image, ET > . . . . .	58
<b>ecvl::DivImpl</b> < Image, T, ET > . . . . .	58
<b>ecvl::DivImpl</b> < T, Image, ET > . . . . .	59
<b>ecvl::Image</b> <b>Image</b> (p. 60) class . . . . .	60
<b>ecvl::Table1D</b> < _StructFun, Args >::integer< i > . . . . .	70
<b>ecvl::SignedTable1D</b> < _StructFun, Args >::integer< i > . . . . .	70
<b>ecvl::Table2D</b> < _StructFun, Args >::integer< i > . . . . .	70
<b>ecvl::Iterator</b> < T > . . . . .	71
<b>MemoryManager</b> . . . . .	74
<b>ecvl::MetaData</b> . . . . .	75
<b>filesystem::path</b> . . . . .	76
<b>ShallowMemoryManager</b> . . . . .	78
<b>ecvl::ShowApp</b> <b>ShowApp</b> (p. 80) is a custom wxApp which allows you to visualize an ECVL <b>Image</b> (p. 60) . . . .	80
<b>ecvl::SignedTable1D</b> < _StructFun, Args > . . . . .	81
<b>ecvl::StructAdd</b> < a, b > . . . . .	83
<b>ecvl::StructCopyImage</b> < SDT, DDT > . . . . .	84
<b>ecvl::StructDiv</b> < a, b, ET > . . . . .	85
<b>ecvl::StructMul</b> < a, b > . . . . .	85
<b>ecvl::StructScalarAdd</b> < DT, T > . . . . .	86
<b>ecvl::StructScalarDiv</b> < DT, T > . . . . .	87
<b>ecvl::StructScalarDivInv</b> < DT, T, ET > In-place division between an <b>Image</b> (p. 60) and a scalar value, without type promotion . . . . .	87
<b>ecvl::StructScalarMul</b> < DT, T > . . . . .	88
<b>ecvl::StructScalarNeg</b> < DT > . . . . .	89

<code>ecvl::StructScalarSub&lt; DT, T &gt;</code>	90
<code>ecvl::StructScalarSubInv&lt; DT, T &gt;</code>	91
<code>ecvl::StructSub&lt; a, b &gt;</code>	91
<code>ecvl::Table1D&lt; _StructFun, Args &gt;</code>	92
<code>ecvl::Table2D&lt; _StructFun, Args &gt;</code>	94
<code>ecvl::TypeInfo&lt; DataType &gt;</code>	96
<code>ecvl::TypeInfo&lt; ecvl::DataType::float32 &gt;</code>	97
<code>ecvl::TypeInfo&lt; ecvl::DataType::float64 &gt;</code>	98
<code>ecvl::TypeInfo&lt; ecvl::DataType::int16 &gt;</code>	98
<code>ecvl::TypeInfo&lt; ecvl::DataType::int32 &gt;</code>	99
<code>ecvl::TypeInfo&lt; ecvl::DataType::int64 &gt;</code>	100
<code>ecvl::TypeInfo&lt; ecvl::DataType::int8 &gt;</code>	100
<code>ecvl::TypeInfo&lt; ecvl::DataType::none &gt;</code>	101
<code>ecvl::TypeInfo&lt; ecvl::DataType::uint16 &gt;</code>	102
<code>ecvl::TypeInfo&lt; ecvl::DataType::uint32 &gt;</code>	102
<code>ecvl::TypeInfo&lt; ecvl::DataType::uint64 &gt;</code>	103
<code>ecvl::TypeInfo&lt; ecvl::DataType::uint8 &gt;</code>	104
<code>ecvl::View&lt; DT &gt;</code>	104
<code>ecvl::wxImagePanel</code>	
WxImagePanel creates a wxPanel to contain an <b>Image</b> (p. 60)	107



## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

<b>arithmetic.cpp</b>	109
<b>arithmetic.h</b>	110
<b>core.cpp</b>	112
<b>core.h</b>	112
<b>datatype.cpp</b>	112
<b>datatype.h</b>	113
<b>datatype_existing_tuples.inc.h</b>	115
<b>datatype_existing_tuples_signed.inc.h</b>	115
<b>datatype_existing_tuples_unsigned.inc.h</b>	115
<b>datatype_matrix.h</b>	115
<b>datatype_tuples.inc.h</b>	116
<b>filesystem.cc</b>	116
<b>filesystem.h</b>	116
<b>gui.cpp</b>	117
<b>gui.h</b>	117
<b>home.h</b>	118
<b>image.cpp</b>	118
<b>image.h</b>	118
<b>imgcodecs.cpp</b>	119
<b>imgcodecs.h</b>	119
<b>imgproc.cpp</b>	120
<b>imgproc.h</b>	121
<b>iterators.h</b>	122
<b>iterators_impl.inc.h</b>	122
<b>memorymanager.cpp</b>	122
<b>memorymanager.h</b>	122
<b>standard_errors.h</b>	123
<b>support_opencv.cpp</b>	125
<b>support_opencv.h</b>	125
<b>test_core.cpp</b>	125



## Chapter 7

# Namespace Documentation

### 7.1 ecvl Namespace Reference

#### Classes

- struct **ConstContiguousIterator**
- class **ConstContiguousView**
- struct **ConstIterator**
- class **ConstView**
- struct **ContiguousIterator**
- class **ContiguousView**
- class **ContiguousViewXYC**
- struct **DivImpl**
- struct **DivImpl**< **Image**, **Image**, **ET** >
- struct **DivImpl**< **Image**, **T**, **ET** >
- struct **DivImpl**< **T**, **Image**, **ET** >
- class **Image**

***Image** (p. 60) class.*

- struct **Iterator**
- class **MetaData**
- class **ShowApp**

***ShowApp** (p. 80) is a custom wxApp which allows you to visualize an ECVL **Image** (p. 60).*

- struct **SignedTable1D**
- struct **StructAdd**
- struct **StructCopyImage**
- struct **StructDiv**
- struct **StructMul**
- struct **StructScalarAdd**
- struct **StructScalarDiv**
- struct **StructScalarDivInv**

*In-place division between an **Image** (p. 60) and a scalar value, without type promotion.*

- struct **StructScalarMul**
- struct **StructScalarNeg**
- struct **StructScalarSub**
- struct **StructScalarSubInv**
- struct **StructSub**
- struct **Table1D**

- struct **Table2D**
- struct **TypeInfo**
- struct **TypeInfo**< **ecvl::DataType::float32** >
- struct **TypeInfo**< **ecvl::DataType::float64** >
- struct **TypeInfo**< **ecvl::DataType::int16** >
- struct **TypeInfo**< **ecvl::DataType::int32** >
- struct **TypeInfo**< **ecvl::DataType::int64** >
- struct **TypeInfo**< **ecvl::DataType::int8** >
- struct **TypeInfo**< **ecvl::DataType::none** >
- struct **TypeInfo**< **ecvl::DataType::uint16** >
- struct **TypeInfo**< **ecvl::DataType::uint32** >
- struct **TypeInfo**< **ecvl::DataType::uint64** >
- struct **TypeInfo**< **ecvl::DataType::uint8** >
- class **View**
- class **wxImagePanel**

**wxImagePanel** (p. 107) creates a **wxPanel** to contain an **Image** (p. 60).

## Enumerations

- enum **DataType** {  
**DataType::ECVL\_TUPLE**, **DataType::int8**, **DataType::int16**, **DataType::int32**,  
**DataType::int64**, **DataType::float32**, **DataType::float64**, **DataType::uint8**,  
**DataType::uint16**, **DataType::uint32**, **DataType::uint64**, **DataType::none** }  
*DataType is an enum class which defines data types allowed for images.*
- enum **ColorType** {  
**ColorType::none**, **ColorType::GRAY**, **ColorType::RGB**, **ColorType::BGR**,  
**ColorType::HSV**, **ColorType::YCbCr** }  
*Enum class representing the ECVL supported color spaces.*
- enum **ThresholdingType** { **ThresholdingType::BINARY**, **ThresholdingType::BINARY\_INV** }  
*Enum class representing the ECVL thresholding types.*
- enum **InterpolationType** {  
**InterpolationType::nearest**, **InterpolationType::linear**, **InterpolationType::area**, **InterpolationType::cubic**,  
**InterpolationType::lanczos4** }  
*Enum class representing the ECVL interpolation types.*

## Functions

- template<DataType ODT, typename IDT >  
**TypeInfo**< ODT >::basetype **saturate\_cast** (IDT v)  
*Saturate a value (of any type) to the specified type.*
- template<typename ODT, typename IDT >  
ODT **saturate\_cast** (const IDT &v)  
*Saturate a value (of any type) to the specified type.*
- void **Add** ( **Image** &src1\_dst, const **Image** &src2)
- void **Sub** ( **Image** &src1\_dst, const **Image** &src2)
- void **Mul** ( **Image** &src1\_dst, const **Image** &src2)
- template<typename T >  
**Image** & **Mul** ( **Image** &img, T value, bool saturate=true)  
*In-place multiplication between an **Image** (p. 60) and a scalar value, without type promotion.*
- template<typename T >  
**Image** & **Mul** (T value, **Image** &img, bool saturate=true)

- `template<typename T >`  
**Image & Add** ( **Image** &img, T value, bool saturate=true)  
*In-place addition between an **Image** (p. 60) and a scalar value, without type promotion.*
- `template<typename T >`  
**Image & Add** (T value, **Image** &img, bool saturate=true)
- `template<typename T >`  
**Image & Sub** ( **Image** &img, T value, bool saturate=true)  
*In-place subtraction between an **Image** (p. 60) and a scalar value, without type promotion.*
- `template<typename T >`  
**Image & Sub** (T value, **Image** &img, bool saturate=true)  
*In-place subtraction between a scalar value and an **Image** (p. 60), without type promotion.*
- **Image & Neg** ( **Image** &img)  
*In-place division between a scalar value and an **Image** (p. 60), without type promotion.*
- `void` **Mul** (const **Image** &src1, const **Image** &src2, **Image** &dst, **DataType** dst\_type, bool saturate=true)  
*Multiplies two Image(s) and stores the result in a third **Image** (p. 60).*
- `void` **Sub** (const **Image** &src1, const **Image** &src2, **Image** &dst, **DataType** dst\_type, bool saturate=true)  
*Subtracts two Image(s) and stores the result in a third **Image** (p. 60).*
- `void` **Add** (const **Image** &src1, const **Image** &src2, **Image** &dst, **DataType** dst\_type, bool saturate=true)  
*Adds two Image(s) and stores the result in a third **Image** (p. 60).*
- `template<typename ET = double>`  
`void` **Div** (const **Image** &src1, const **Image** &src2, **Image** &dst, **DataType** dst\_type, bool saturate=true, ET epsilon=std::numeric\_limits< double >::min())  
*Divides two Image(s) and stores the result in a third **Image** (p. 60).*
- `template<typename ST1 , typename ST2 , typename ET = double>`  
`constexpr` **Image & Div** (const ST1 &src1, const ST2 &src2, **Image** &dst, bool saturate=true, ET epsilon=std::numeric\_limits< double >::min())
- `uint8_t` **DataTypeSize** ( **DataType** dt)  
*Provides the size in bytes of a given DataType.*
- `constexpr` `size_t` **DataTypeSize** ()  
*Function to get the number of existing DataType at compile time.*
- `constexpr` `size_t` **DataTypeSignedSize** ()  
*Function to get the number of existing signed DataType at compile time.*
- `constexpr` `std::array< DataType, DataTypeSize()>` **DataTypeArray** ()  
*Function to get a std::array with all the DataType values at compile time.*
- `constexpr` `std::array< DataType, DataTypeSignedSize()>` **DataTypeSignedArray** ()  
*Function to get a std::array with all the signed DataType values at compile time.*
- `void` **RearrangeChannels** (const **Image** &src, **Image** &dst, const `std::string` &channels)  
*Changes the order of the **Image** (p. 60) dimensions.*
- `void` **CopyImage** (const **Image** &src, **Image** &dst, **DataType** new\_type= **DataType::none**)  
*Copies the source **Image** (p. 60) into the destination **Image** (p. 60).*
- `bool` **ImRead** (const `std::string` &filename, **Image** &dst)  
*Loads an image from a file.*
- `bool` **ImRead** (const `filesystem::path` &filename, **Image** &dst)
- `bool` **ImWrite** (const `std::string` &filename, const **Image** &src)  
*Saves an image into a specified file.*
- `bool` **ImWrite** (const `filesystem::path` &filename, const **Image** &src)
- `void` **ResizeDim** (const **ecvl::Image** &src, **ecvl::Image** &dst, const `std::vector< int >` &newdims, **InterpolationType** interp= **InterpolationType::linear**)  
*Resizes an **Image** (p. 60) to a new dimension.*
- `void` **ResizeScale** (const **ecvl::Image** &src, **ecvl::Image** &dst, const `std::vector< double >` &scales, **InterpolationType** interp= **InterpolationType::linear**)  
*Resizes an **Image** (p. 60) by scaling the dimentions to a given scale factor.*

- void **Flip2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Flips an **Image** (p. 60).*
- void **Mirror2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Mirrors an **Image** (p. 60).*
- void **Rotate2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, const std::vector< double > &center={}, double scale=1.0, **InterpolationType** interp= **InterpolationType::linear**)  
*Rotates an **Image** (p. 60).*
- void **RotateFullImage2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, double scale=1.0, **InterpolationType** interp= **InterpolationType::linear**)  
*Rotates an **Image** (p. 60) resizing the output accordingly.*
- void **ChangeColorSpace** (const **Image** &src, **Image** &dst, **ColorType** new\_type)  
*Copies the source **Image** (p. 60) into destination **Image** (p. 60) changing the color space.*
- void **Threshold** (const **Image** &src, **Image** &dst, double thresh, double maxval, **ThresholdingType** thresh\_type= **ThresholdingType::BINARY**)  
*Applies a fixed threshold to an input **Image** (p. 60).*
- double **OtsuThreshold** (const **Image** &src)  
*Calculates the Otsu thresholding value.*
- **ecvl::Image MatToImage** (const cv::Mat &m)  
*Convert a cv::Mat into an **ecvl::Image** (p. 60).*
- cv::Mat **ImageToMat** (const **Image** &img)  
*Convert an ECVL **Image** (p. 60) into OpenCV Mat.*
- void **ImShow** (const **Image** &img)  
*Displays an **Image** (p. 60).*
- wxImage **wx\_from\_mat** ( **Image** &img)

## 7.1.1 Enumeration Type Documentation

### 7.1.1.1 ColorType

enum **ecvl::ColorType** [strong]

Enum class representing the ECVL supported color spaces.

#### Enumerator

##### Enumerator

none	Special ColorType for Images that contain only data and do not have any ColorType
GRAY	Gray-scale ColorType
RGB	RGB ColorType
BGR	BGR ColorType
HSV	HSV ColorType
YCbCr	YCbCr ColorType

Definition at line 27 of file image.h.

## 7.1.1.2 DataType

```
enum ecvl::DataType [strong]
```

DataType is an enum class which defines data types allowed for images.

## Enumerator

## Enumerator

ECVL_TUPLE	
int8	int8_t
int16	int16_t
int32	int32_t
int64	int64_t
float32	float
float64	double
uint8	uint8_t
uint16	uint16_t
uint32	uint32_t
uint64	uint64_t
none	none type

Definition at line 15 of file datatype.h.

## 7.1.1.3 InterpolationType

```
enum ecvl::InterpolationType [strong]
```

Enum class representing the ECVL interpolation types.

## Enumerator

## Enumerator

nearest	Nearest neighbor interpolation
linear	Bilinear interpolation
area	Resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire-free results. But when the image is zoomed, it is similar to the nearest method.
cubic	Bicubic interpolation
lanczos4	Lanczos interpolation over 8x8 neighborhood

Definition at line 23 of file imgproc.h.

#### 7.1.1.4 ThresholdingType

```
enum ecvl::ThresholdingType [strong]
```

Enum class representing the ECVL threhsolding types.

##### Enumerator

##### Enumerator

BINARY	$\text{dst}(x,y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x,y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
BINARY_INV	$\text{dst}(x,y) = \begin{cases} 0 & \text{if } \text{src}(x,y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$

Definition at line 14 of file imgproc.h.

### 7.1.2 Function Documentation

#### 7.1.2.1 Add() [1/4]

```
void ecvl::Add (
    Image & src1_dst,
    const Image & src2 )
```

Definition at line 36 of file arithmetic.cpp.

#### 7.1.2.2 Add() [2/4]

```
template<typename T >
Image& ecvl::Add (
    Image & img,
    T value,
    bool saturate = true )
```

In-place addition between an **Image** (p. 60) and a scalar value, without type promotion.

The **Add()** (p. 18) function sums a scalar value to the input **Image** (p. 60) and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

##### Parameters

##### Parameters

in, out	<i>img</i>	<b>Image</b> (p. 60) to be summed (in-place) by a scalar value.
in	<i>value</i>	Scalar value to use for the sum.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.



**Returns**

Reference to the **Image** (p. 60) containing the result of the sum.

Definition at line 156 of file arithmetic.h.

**7.1.2.3 Add()** [3/4]

```
template<typename T >
Image& ecvl::Add (
    T value,
    Image & img,
    bool saturate = true )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 166 of file arithmetic.h.

**7.1.2.4 Add()** [4/4]

```
void ecvl::Add (
    const Image & src1,
    const Image & src2,
    Image & dst,
    DataType dst_type,
    bool saturate = true )
```

Adds two **Image**(s) and stores the result in a third **Image** (p. 60).

This procedure adds src1 and src2 **Image**(s) (src1 + src2) and stores the result in the dst **Image** (p. 60) that will have the specified **DataType**. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

**Parameters****Parameters**

in	<i>src1</i>	Augend (first addend) <b>Image</b> (p. 60).
in	<i>src2</i>	Addend (second addend) <b>Image</b> (p. 60).
out	<i>dst</i>	<b>Image</b> (p. 60) into which save the result of the division.
in	<i>dst_type</i>	<b>DataType</b> that destination <b>Image</b> (p. 60) must have at the end of the operation.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

**Returns**

Definition at line 126 of file arithmetic.cpp.

### 7.1.2.5 ChangeColorSpace()

```
void ecvl::ChangeColorSpace (
    const Image & src,
    Image & dst,
    ColorType new_type )
```

Copies the source **Image** (p. 60) into destination **Image** (p. 60) changing the color space.

The ChangeColorSpace procedure convert the color space of the source **Image** (p. 60) into the specified color space. New data are copied into destination **Image** (p. 60). Source and destination can be contiguous or not and can also be the same **Image** (p. 60).

#### Parameters

##### Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60) to convert in the new color space.
out	<i>dst</i>	The output <b>Image</b> (p. 60) in the "new_type" color space.
in	<i>new_type</i>	The new color space in which the src <b>Image</b> (p. 60) must be converted.

Definition at line 168 of file imgproc.cpp.

### 7.1.2.6 CopyImage()

```
void ecvl::CopyImage (
    const Image & src,
    Image & dst,
    DataType new_type = DataType::none )
```

Copies the source **Image** (p. 60) into the destination **Image** (p. 60).

The **CopyImage()** (p. 20) procedure takes an **Image** (p. 60) and copies its data into the destination **Image** (p. 60). Source and destination cannot be the same **Image** (p. 60). Source cannot be a **Image** (p. 60) with **DataType::none** (p. 17). The optional new\_type parameter can be used to change the DataType of the destination **Image** (p. 60). This function is mainly designed to change the DataType of an **Image** (p. 60), copying its data into a new **Image** (p. 60) or to copy an **Image** (p. 60) into a **View** (p. 104) as a patch. So if you just want to copy an **Image** (p. 60) as it is, use the copy constructor or = instead. Anyway, the procedure will handle all the possible situations that may happen trying to avoid unnecessary allocations. When the DataType is not specified the function will have the following behaviors:

- if the destination **Image** (p. 60) is empty the source will be directly copied into the destination.
- if source and destination have different size in memory or different channels and the destination is the owner of data, the procedure will overwrite the destination **Image** (p. 60) creating a new **Image** (p. 60) (channels and dimensions will be the same of the source **Image** (p. 60), pixels type (DataType) will be the same of the destination **Image** (p. 60) if they are not none or the same of the source otherwise).
- if source and destination have different size in memory or different channels and the destination is not the owner of data, the procedure will throw an exception.
- if source and destination have different color types and the destination is the owner of data, the procedure produces a destination **Image** (p. 60) with the same color type of the source.
- if source and destination have different color types and the destination is not the owner of data, the procedure will throw an exception. When the DataType is specified the function will have the same behavior, but the destination **Image** (p. 60) will have the specified DataType.

**Parameters****Parameters**

in	<i>src</i>	Source <b>Image</b> (p. 60) to be copied into destination <b>Image</b> (p. 60).
out	<i>dst</i>	Destination <b>Image</b> (p. 60) that will hold a copy of the source <b>Image</b> (p. 60). Cannot be the source <b>Image</b> (p. 60).
in	<i>new_type</i>	Desired type for the destination <b>Image</b> (p. 60) after the copy. If none (default) the destination <b>Image</b> (p. 60) will preserve its type if it is not empty, otherwise it will have the same type of the source <b>Image</b> (p. 60).

Definition at line 95 of file image.cpp.

**7.1.2.7 DataTypeArray()**

```
constexpr std::array< DataType, DataTypeSize()> ecvl::DataTypeArray ( )
```

Function to get a std::array with all the **DataType** values at compile time.

**Returns**

A std::array with all the **DataType** values.

Definition at line 71 of file datatype.h.

**7.1.2.8 DataTypeSignedArray()**

```
constexpr std::array< DataType, DataTypeSignedSize()> ecvl::DataTypeSignedArray ( )
```

Function to get a std::array with all the signed **DataType** values at compile time.

**Returns**

A std::array with all the signed **DataType** values.

Definition at line 88 of file datatype.h.

**7.1.2.9 DataTypeSignedSize()**

```
constexpr size_t ecvl::DataTypeSignedSize ( )
```

Function to get the number of existing signed **DataType** at compile time.

**Returns**

The number of existing signed **DataType**.

Definition at line 16 of file datatype.h.

**7.1.2.10 DataTypeSize()** [1/2]

```
uint8_t ecvl::DataTypeSize (
    DataType dt )
```

Provides the size in bytes of a given **DataType**.

Given one of the **DataType** (p. 17), the function returns its size in bytes.

**Parameters****Parameters**

in	<i>dt</i>	A <b>DataType</b> .
----	-----------	---------------------

**Returns**

The **DataType** size in bytes

Definition at line 11 of file datatype.cpp.

**7.1.2.11 DataTypeSize()** [2/2]

```
constexpr size_t ecvl::DataTypeSize ( )
```

Function to get the number of existing **DataType** at compile time.

**Returns**

The number of existing **DataType**.

Definition at line 43 of file datatype.h.

**7.1.2.12 Div()** [1/2]

```
template<typename ET = double>
void ecvl::Div (
    const Image & src1,
    const Image & src2,
    Image & dst,
    DataType dst_type,
    bool saturate = true,
    ET epsilon = std::numeric_limits<double>::min() )
```

Divides two **Image**(s) and stores the result in a third **Image** (p. 60).

This procedure divides the src1 **Image** (p. 60) by the src2 **Image** (p. 60) (src1/src2) and stores the result into the dst **Image** (p. 60) that will have the specified **DataType**. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

**Parameters****Parameters**

in	<i>src1</i>	Dividend (numerator) <b>Image</b> (p. 60).
in	<i>src2</i>	Divisor (denominator) <b>Image</b> (p. 60).
out	<i>dst</i>	<b>Image</b> (p. 60) into which save the result of the division.
in	<i>dst_type</i>	<b>DataType</b> that destination <b>Image</b> (p. 60) must have at the end of the operation.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.
in	<i>epsilon</i>	Small value to be added to the <b>Image</b> (p. 60) values before performing the division. If not specified by default it is the minimum positive number representable in a double.

**Returns**

Definition at line 434 of file arithmetic.h.

**7.1.2.13 Div()** [2/2]

```
template<typename ST1 , typename ST2 , typename ET = double>
constexpr Image& ecvl::Div (
    const ST1 & src1,
    const ST2 & src2,
    Image & dst,
    bool saturate = true,
    ET epsilon = std::numeric_limits<double>::min() )
```

Definition at line 516 of file arithmetic.h.

**7.1.2.14 Flip2D()**

```
void ecvl::Flip2D (
    const ecvl::Image & src,
    ecvl::Image & dst )
```

Flips an **Image** (p. 60).

The Flip2D procedure vertically flips an **Image** (p. 60).

**Parameters****Parameters**

in	src	The input <b>Image</b> (p. 60).
out	dst	The output flipped <b>Image</b> (p. 60).

Definition at line 73 of file imgproc.cpp.

**7.1.2.15 ImageToMat()**

```
cv::Mat ecvl::ImageToMat (
    const Image & img )
```

Convert an ECVL **Image** (p. 60) into OpenCV Mat.

**Parameters****Parameters**

in	img	Input ECVL <b>Image</b> (p. 60).
----	-----	----------------------------------

**Returns**

Output OpenCV Mat.

Definition at line 98 of file support\_opencv.cpp.

7.1.2.16 `ImRead()` [1/2]

```
bool ecv1::ImRead (
    const std::string & filename,
    Image & dst )
```

Loads an image from a file.

The function `ImRead` loads an image from the specified file. If the image cannot be read for any reason, the function creates an empty **Image** (p. 60) and returns false.

## Parameters

## Parameters

in	<i>filename</i>	A <code>std::string</code> identifying the file name. In order to be platform independent consider to use <b><code>ImRead(const filesystem::path&amp; filename, Image&amp; dst)</code></b> (p. 24) .
out	<i>dst</i>	<b>Image</b> (p. 60) in which data will be stored.

## Returns

true if the image is correctly read, false otherwise.

Definition at line 10 of file `imgcodecs.cpp`.

7.1.2.17 `ImRead()` [2/2]

```
bool ecv1::ImRead (
    const filesystem::path & filename,
    Image & dst )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

This variant of `ImRead` is platform independent.

## Parameters

## Parameters

in	<i>filename</i>	A <b><code>filesystem::path</code></b> (p. 76) identifying the file name.
out	<i>dst</i>	<b>Image</b> (p. 60) in which data will be stored.

## Returns

true if the image is correctly read, false otherwise.

Definition at line 16 of file `imgcodecs.cpp`.

## 7.1.2.18 ImShow()

```
void ecvl::ImShow (
    const Image & img )
```

Displays an **Image** (p. 60).

The ImShow function instantiates a **ShowApp** (p. 80) and starts it with a wxEntry() call. The image is shown with its original size.

## Parameters

## Parameters

in	<i>img</i>	<b>Image</b> (p. 60) to be shown.
----	------------	-----------------------------------

Definition at line 87 of file gui.cpp.

## 7.1.2.19 ImWrite() [1/2]

```
bool ecvl::ImWrite (
    const std::string & filename,
    const Image & src )
```

Saves an image into a specified file.

The function ImWrite saves the input image into a specified file. The image format is chosen based on the filename extension. The following sample shows how to create a BGR image and save it to the PNG file "test.png":

```
#include "ecvl/core.h"
using namespace std;
using namespace ecvl;
using namespace filesystem;
int main()
{
    // Create BGR Image
    Image img({ 500, 500, 3 }, DataType::uint8, "xyz", ColorType::BGR);

    // Populate Image with pseudo-random data
    for (int r = 0; r < img.dims_[1]; ++r) {
        for (int c = 0; c < img.dims_[0]; ++c) {
            *img.Ptr({ c, r, 0 }) = 255;
            *img.Ptr({ c, r, 1 }) = (r / 2) % 255;
            *img.Ptr({ c, r, 2 }) = (r / 2) % 255;
        }
    }
    ImWrite(path("./test.png"), img);
    return EXIT_SUCCESS;
}
```

## Parameters

## Parameters

in	<i>filename</i>	A std::string identifying the output file name. In order to be platform independent consider to use <b>ImWrite(const filesystem::path&amp; filename, const Image&amp; src)</b> (p. 26).
in	<i>src</i>	<b>Image</b> (p. 60) to be saved.

## Returns

true if the image is correctly write, false otherwise.

Definition at line 21 of file imgcodecs.cpp.

### 7.1.2.20 ImWrite() [2/2]

```
bool ecvl::ImWrite (
    const filesystem::path & filename,
    const Image & src )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

This variant of ImWrite is platform independent.

#### Parameters

##### Parameters

in	<i>filename</i>	A <b>filesystem::path</b> (p. 76) identifying the output file name.
in	<i>src</i>	<b>Image</b> (p. 60) to be saved.

#### Returns

true if the image is correctly write, false otherwise.

Definition at line 26 of file imgcodecs.cpp.

### 7.1.2.21 MatToImage()

```
Image ecvl::MatToImage (
    const cv::Mat & m )
```

Convert a cv::Mat into an **ecvl::Image** (p. 60).

#### Parameters

##### Parameters

in	<i>m</i>	Input OpenCV Mat.
----	----------	-------------------

#### Returns

ECVL image.

Definition at line 7 of file support\_opencv.cpp.

### 7.1.2.22 Mirror2D()

```
void ecvl::Mirror2D (
    const ecvl::Image & src,
    ecvl::Image & dst )
```

Mirrors an **Image** (p. 60).

The Mirror2D procedure horizontally flips an **Image** (p. 60).

#### Parameters

##### Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60).
out	<i>dst</i>	The output mirrored <b>Image</b> (p. 60).

Definition at line 89 of file imgproc.cpp.



7.1.2.23 `Mul()` [1/4]

```
void ecvl::Mul (
    Image & src1_dst,
    const Image & src2 )
```

Definition at line 70 of file arithmetic.cpp.

7.1.2.24 `Mul()` [2/4]

```
template<typename T >
Image& ecvl::Mul (
    Image & img,
    T value,
    bool saturate = true )
```

In-place multiplication between an **Image** (p. 60) and a scalar value, without type promotion.

The **Mul()** (p. 26) function multiplies an input image by a scalar value and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

## Parameters

## Parameters

in, out	<i>img</i>	<b>Image</b> (p. 60) to be multiplied (in-place) by a scalar value.
in	<i>value</i>	Scalar value to use for the multiplication.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

## Returns

Reference to the **Image** (p. 60) containing the result of the multiplication.

Definition at line 107 of file arithmetic.h.

7.1.2.25 `Mul()` [3/4]

```
template<typename T >
Image& ecvl::Mul (
    T value,
    Image & img,
    bool saturate = true )
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 117 of file arithmetic.h.

### 7.1.2.26 Mul() [4/4]

```
void ecvl::Mul (
    const Image & src1,
    const Image & src2,
    Image & dst,
    DataType dst_type,
    bool saturate = true )
```

Multiplies two **Image**(s) and stores the result in a third **Image** (p. 60).

This procedure multiplies two **Image**(s) together and stores the result in a third **Image** (p. 60) that will have the specified **DataType**. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

#### Parameters

##### Parameters

in	<i>src1</i>	Multiplier (first factor) <b>Image</b> (p. 60).
in	<i>src2</i>	Multiplicand (second factor) <b>Image</b> (p. 60).
out	<i>dst</i>	<b>Image</b> (p. 60) into which save the result of the multiplication.
in	<i>dst_type</i>	<b>DataType</b> that destination <b>Image</b> (p. 60) must have at the end of the operation.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

#### Returns

Definition at line 125 of file arithmetic.cpp.

### 7.1.2.27 Neg()

```
Image & ecvl::Neg (
    Image & img )
```

In-place division between a scalar value and an **Image** (p. 60), without type promotion.

The **Div()** (p. 22) function divides a scalar value by the input **Image** (p. 60) and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

#### Parameters

##### Parameters

in	<i>value</i>	Scalar value to use for the division (Dividend).
in, out	<i>img</i>	Divisor of the operation. It will store the final result.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.
in	<i>epsilon</i>	Small value to be added to the <b>Image</b> (p. 60) values before performing the division. If not specified by default it is the minimum positive number representable in a double.

**Returns**

Reference to the **Image** (p. 60) containing the result of the division. In-place negation of an **Image** (p. 60). The **Neg()** (p. 28) function negates every value of an **Image** (p. 60), and stores the the result in the same image. The type of the image will not change.

**Parameters****Parameters**

<i>in, out</i>	<i>img</i>	<b>Image</b> (p. 60) to be negated (in-place).
----------------	------------	--

**Returns**

Reference to the **Image** (p. 60) containing the result of the negation.

Definition at line 93 of file arithmetic.cpp.

**7.1.2.28 OtsuThreshold()**

```
double ecvl::OtsuThreshold (
    const Image & src )
```

Calculates the Otsu thresholding value.

The OtsuThreshold function calculates the Otsu threshold value over a given input **Image** (p. 60). the **Image** (p. 60) must be **ColorType::GRAY** (p. 16).

**Parameters****Parameters**

<i>in</i>	<i>src</i>	Input <b>Image</b> (p. 60) on which to calculate the Otsu threshold value.
-----------	------------	--

**Returns**

Otsu threshold value.

Definition at line 274 of file imgproc.cpp.

**7.1.2.29 RearrangeChannels()**

```
void ecvl::RearrangeChannels (
    const Image & src,
    Image & dst,
    const std::string & channels )
```

Changes the order of the **Image** (p. 60) dimensions.

The RearrangeChannels procedure changes the order of the input **Image** (p. 60) dimensions saving the result into the output **Image** (p. 60). The new order of dimensions can be specified as a string through the "channels" parameter. Input and output Images can be the same. The number of channels of the input **Image** (p. 60) must be the same of required channels.

**Parameters****Parameters**

<i>in</i>	<i>src</i>	Input <b>Image</b> (p. 60) on which to rearrange dimensions.
<i>out</i>	<i>dst</i>	The output rearranged <b>Image</b> (p. 60). Can be the src <b>Image</b> (p. 60).
<i>in</i>	<i>channels</i>	Desired order of <b>Image</b> (p. 60) channels.

Definition at line 49 of file image.cpp.

## 7.1.2.30 ResizeDim()

```
void ecv1::ResizeDim (
    const   ecv1::Image & src,
    ecv1::Image & dst,
    const std::vector< int > & newdims,
    InterpolationType interp = InterpolationType::linear )
```

Resizes an **Image** (p. 60) to a new dimension.

The function resizes **Image** (p. 60) *src* and outputs the result in *dst*.

## Parameters

## Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60).
out	<i>dst</i>	The output resized <b>Image</b> (p. 60).
in	<i>newdims</i>	std::vector<int> that specifies the new size of each dimension. The vector size must match the <i>src</i> <b>Image</b> (p. 60) dimensions, excluding the color channel
in	<i>interp</i>	InterpolationType to be used. See <b>InterpolationType</b> (p. 17).

Definition at line 30 of file imgproc.cpp.

## 7.1.2.31 ResizeScale()

```
void ecv1::ResizeScale (
    const   ecv1::Image & src,
    ecv1::Image & dst,
    const std::vector< double > & scales,
    InterpolationType interp = InterpolationType::linear )
```

Resizes an **Image** (p. 60) by scaling the dimensions to a given scale factor.

The function resizes **Image** (p. 60) *src* and outputs the result in *dst*.

## Parameters

## Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60).
out	<i>dst</i>	The output resized <b>Image</b> (p. 60).
in	<i>scales</i>	std::vector<double> that specifies the scale to apply to each dimension. The vector size must match the <i>src</i> <b>Image</b> (p. 60) dimensions, excluding the color channel.
in	<i>interp</i>	InterpolationType to be used. See <b>InterpolationType</b> (p. 17).

Definition at line 50 of file imgproc.cpp.

## 7.1.2.32 Rotate2D()

```
void ecvl::Rotate2D (
    const ecvl::Image & src,
    ecvl::Image & dst,
    double angle,
    const std::vector< double > & center = {},
    double scale = 1.0,
    InterpolationType interp = InterpolationType::linear )
```

Rotates an **Image** (p. 60).

The Rotate2D procedure rotates an **Image** (p. 60) of a given angle (expressed in degrees) in a clockwise manner, with respect to a given center. The value of unknown pixels in the output **Image** (p. 60) are set to 0. The output **Image** (p. 60) is guaranteed to have the same dimensions as the input one. An optional scale parameter can be provided: this won't change the output **Image** (p. 60) size, but the image is scaled during rotation. Different interpolation types are available, see **InterpolationType** (p. 17).

## Parameters

## Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60).
out	<i>dst</i>	The output rotated <b>Image</b> (p. 60).
in	<i>angle</i>	The rotation angle in degrees.
in	<i>center</i>	A std::vector<double> representing the coordinates of the rotation center. If empty, the center of the image is used.
in	<i>scale</i>	Optional scaling factor.
in	<i>interp</i>	Interpolation type used. Default is <b>InterpolationType::linear</b> (p. 17).

Definition at line 105 of file imgproc.cpp.

## 7.1.2.33 RotateFullImage2D()

```
void ecvl::RotateFullImage2D (
    const ecvl::Image & src,
    ecvl::Image & dst,
    double angle,
    double scale = 1.0,
    InterpolationType interp = InterpolationType::linear )
```

Rotates an **Image** (p. 60) resizing the output accordingly.

The RotateFullImage2D procedure rotates an **Image** (p. 60) of a given angle (expressed in degrees) in a clockwise manner. The value of unknown pixels in the output **Image** (p. 60) are set to 0. The output **Image** (p. 60) is guaranteed to contain all the pixels of the rotated image. Thus, its dimensions can be different from those of the input. An optional scale parameter can be provided. Different interpolation types are available, see **InterpolationType** (p. 17).

## Parameters

## Parameters

in	<i>src</i>	The input <b>Image</b> (p. 60).
out	<i>dst</i>	The rotated output <b>Image</b> (p. 60).
in	<i>angle</i>	The rotation angle in degrees.
in	<i>scale</i>	Optional scaling factor.
in	<i>interp</i>	Interpolation type used. Default is <b>InterpolationType::linear</b> (p. 17).

Definition at line 134 of file imgproc.cpp.

**7.1.2.34 saturate\_cast()** [1/2]

```
template<DataType ODT, typename IDT >
TypeInfo<ODT>::basetype ecvl::saturate_cast (
    IDT v )
```

Saturate a value (of any type) to the specified type.

Given an input of any type the saturate\_cast function provide an output return value of the specified type applying saturation. When the input value in greater than the maximum possible value (max) for the output type, the max value is returned. When the input value in lower than the minimum possible value (min) for the output type, the min value is returned.

**Parameters****Parameters**

in	v	Input value (of any type).
----	---	----------------------------

**Returns**

Input value after cast and saturation.

Definition at line 27 of file arithmetic.h.

**7.1.2.35 saturate\_cast()** [2/2]

```
template<typename ODT , typename IDT >
ODT ecvl::saturate_cast (
    const IDT & v )
```

Saturate a value (of any type) to the specified type.

Given an input of any type the saturate\_cast function provide an output return value of the specified type applying saturation. When the input value in greater than the maximum possible value (max) for the output type, the max value is returned. When the input value in lower than the minimum possible value (min) for the output type, the min value is returned.

**Parameters****Parameters**

in	v	Input value (of any type).
----	---	----------------------------

**Returns**

Input value after cast and saturation.

Definition at line 54 of file arithmetic.h.

**7.1.2.36 Sub()** [1/4]

```
void ecvl::Sub (
    Image & src1_dst,
    const Image & src2 )
```

Definition at line 53 of file arithmetic.cpp.

**7.1.2.37 Sub()** [2/4]

```
template<typename T >
Image& ecvl::Sub (
    Image & img,
    T value,
    bool saturate = true )
```

In-place subtraction between an **Image** (p. 60) and a scalar value, without type promotion.

The **Sub()** (p. 32) function subtracts a scalar value from the input **Image** (p. 60) and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

**Parameters****Parameters**

in, out	<i>img</i>	<b>Image</b> (p. 60) to be subtracted (in-place) by a scalar value.
in	<i>value</i>	Scalar value to use for the subtraction.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

**Returns**

Reference to the **Image** (p. 60) containing the result of the subtraction.

Definition at line 205 of file arithmetic.h.

**7.1.2.38 Sub()** [3/4]

```
template<typename T >
Image& ecvl::Sub (
    T value,
    Image & img,
    bool saturate = true )
```

In-place subtraction between a scalar value and an **Image** (p. 60), without type promotion.

The **Sub()** (p. 32) function subtracts the input **Image** (p. 60) from a scalar value and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

**Parameters****Parameters**

in	<i>value</i>	Scalar value to use for the subtraction (Minuend).
in, out	<i>img</i>	Subtrahend of the operation. It will store the final result.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

**Returns**

Reference to the **Image** (p. 60) containing the result of the subtraction.

Definition at line 248 of file arithmetic.h.

**7.1.2.39 Sub()** [4/4]

```
void ecvl::Sub (
    const Image & src1,
    const Image & src2,
    Image & dst,
    DataType dst_type,
    bool saturate = true )
```

Subtracts two **Image**(s) and stores the result in a third **Image** (p. 60).

This procedure subtracts the src2 **Image** (p. 60) from the src1 **Image** (p. 60) (src1 - src2) and stores the result in the dst **Image** (p. 60) that will have the specified **DataType**. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

**Parameters****Parameters**

in	<i>src1</i>	Minuend <b>Image</b> (p. 60).
in	<i>src2</i>	Subtrahend <b>Image</b> (p. 60).
out	<i>dst</i>	<b>Image</b> (p. 60) into which save the result of the division.
in	<i>dst_type</i>	<b>DataType</b> that destination <b>Image</b> (p. 60) must have at the end of the operation.
in	<i>saturate</i>	Whether to apply saturation or not. Default is true.

**Returns**

Definition at line 127 of file arithmetic.cpp.

**7.1.2.40 Threshold()**

```
void ecvl::Threshold (
    const Image & src,
    Image & dst,
    double thresh,
    double maxval,
    ThresholdingType thresh_type = ThresholdingType::BINARY )
```

Applies a fixed threshold to an input **Image** (p. 60).

The Threshold function applies a fixed thresholding to an input **Image** (p. 60). The function is useful to get a binary image out of a grayscale (**ColorType::GRAY** (p. 16)) **Image** (p. 60) or to remove noise filtering out pixels with too small or too large values. Anyway, the function can be applied to any input **Image** (p. 60). The pixels up to "thresh" value will be set to 0, the pixels above this value will be set to "maxvalue" if "thresh\_type" is **ThresholdingType::BINARY** (p. 18) (default). The opposite will happen if "thresh\_type" is **ThresholdingType::BINARY\_INV** (p. 18).



**Bug** Input and output Images may have different color spaces.

#### Parameters

##### Parameters

in	<i>src</i>	Input <b>Image</b> (p. 60) on which to apply the threshold.
out	<i>dst</i>	The output thresholded <b>Image</b> (p. 60).
in	<i>thresh</i>	Threshold value.
in	<i>maxval</i>	The maximum values in the thresholded <b>Image</b> (p. 60).
in	<i>thresh_type</i>	Type of threshold to be applied, see <b>ThresholdingType</b> (p. 18). The default value is <b>ThresholdingType::BINARY</b> (p. 18).

Definition at line 258 of file imgproc.cpp.

#### 7.1.2.41 wx\_from\_mat()

```
wxImage ecv1::wx_from_mat (
    Image & img )
```

Definition at line 45 of file gui.cpp.

## 7.2 filesystem Namespace Reference

### Classes

- class **path**

### Functions

- **path operator/** (const **path** &lhs, const **path** &rhs)
- bool **exists** (const **path** &p)
- bool **exists** (const **path** &p, std::error\_code &ec)
- bool **create\_directories** (const **path** &p)
- bool **create\_directories** (const **path** &p, std::error\_code &ec)
- void **copy** (const **path** &from, const **path** &to)
- void **copy** (const **path** &from, const **path** &to, std::error\_code &ec)
- bool **exists** (const **path** &p, error\_code &ec)
- bool **create\_directories** (const **path** &p, error\_code &ec)
- void **copy** (const **path** &from, const **path** &to, error\_code &ec)

#### 7.2.1 Function Documentation

#### 7.2.1.1 `copy()` [1/3]

```
void filesystem::copy (
    const path & from,
    const path & to,
    error_code & ec )
```

Definition at line 93 of file filesystem.cc.

#### 7.2.1.2 `copy()` [2/3]

```
void filesystem::copy (
    const path & from,
    const path & to )
```

Definition at line 77 of file filesystem.cc.

#### 7.2.1.3 `copy()` [3/3]

```
void filesystem::copy (
    const path & from,
    const path & to,
    std::error_code & ec )
```

#### 7.2.1.4 `create_directories()` [1/3]

```
bool filesystem::create_directories (
    const path & p,
    error_code & ec )
```

Definition at line 61 of file filesystem.cc.

#### 7.2.1.5 `create_directories()` [2/3]

```
bool filesystem::create_directories (
    const path & p )
```

Definition at line 43 of file filesystem.cc.

#### 7.2.1.6 create\_directories() [3/3]

```
bool filesystem::create_directories (
    const path & p,
    std::error_code & ec )
```

#### 7.2.1.7 exists() [1/3]

```
bool filesystem::exists (
    const path & p,
    error_code & ec )
```

Definition at line 38 of file filesystem.cc.

#### 7.2.1.8 exists() [2/3]

```
bool filesystem::exists (
    const path & p )
```

Definition at line 20 of file filesystem.cc.

#### 7.2.1.9 exists() [3/3]

```
bool filesystem::exists (
    const path & p,
    std::error_code & ec )
```

#### 7.2.1.10 operator/()

```
path filesystem::operator/ (
    const path & lhs,
    const path & rhs ) [inline]
```

Definition at line 109 of file filesystem.h.



## Chapter 8

# Class Documentation

### 8.1 `ecvl::ConstContiguousIterator< T >` Struct Template Reference

```
#include <iterators.h>
```

#### Public Member Functions

- **ConstContiguousIterator** (const **Image** &img, std::vector< int > pos={})
- **ConstContiguousIterator** & **operator++** ()
- const T & **operator\*** () const
- const T \* **operator->** () const
- bool **operator==** (const **ConstContiguousIterator** &rhs) const
- bool **operator!=** (const **ConstContiguousIterator** &rhs) const

#### Public Attributes

- uint8\_t \* **ptr\_**
- const **Image** \* **img\_**

#### 8.1.1 Detailed Description

```
template<typename T>  
struct ecvl::ConstContiguousIterator< T >
```

Definition at line 68 of file iterators.h.

#### 8.1.2 Constructor & Destructor Documentation

### 8.1.2.1 ConstContiguousIterator()

```
template<typename T >
ConstContiguousIterator::ConstContiguousIterator (
    const Image & img,
    std::vector< int > pos = {} )
```

Definition at line 78 of file image.h.

## 8.1.3 Member Function Documentation

### 8.1.3.1 operator\*()

```
template<typename T >
const T& ecvl::ConstContiguousIterator< T >::operator * ( ) const [inline]
```

Definition at line 74 of file iterators.h.

### 8.1.3.2 operator!=(())

```
template<typename T >
bool ecvl::ConstContiguousIterator< T >::operator!= (
    const ConstContiguousIterator< T > & rhs ) const [inline]
```

Definition at line 77 of file iterators.h.

### 8.1.3.3 operator++()

```
template<typename T >
ConstContiguousIterator& ecvl::ConstContiguousIterator< T >::operator++ ( ) [inline]
```

Definition at line 73 of file iterators.h.

### 8.1.3.4 operator->()

```
template<typename T >
const T* ecvl::ConstContiguousIterator< T >::operator-> ( ) const [inline]
```

Definition at line 75 of file iterators.h.

## 8.1.3.5 operator==()

```
template<typename T >
bool  ecvl::ConstContiguousIterator< T >::operator== (
    const  ConstContiguousIterator< T > & rhs ) const  [inline]
```

Definition at line 76 of file iterators.h.

## 8.1.4 Member Data Documentation

## 8.1.4.1 img\_

```
template<typename T >
const  Image*  ecvl::ConstContiguousIterator< T >::img_
```

Definition at line 70 of file iterators.h.

## 8.1.4.2 ptr\_

```
template<typename T >
uint8_t*  ecvl::ConstContiguousIterator< T >::ptr_
```

Definition at line 69 of file iterators.h.

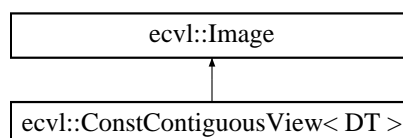
The documentation for this struct was generated from the following files:

- iterators.h
- image.h
- iterators\_impl.inc.h

## 8.2 ecvl::ConstContiguousView&lt; DT &gt; Class Template Reference

```
#include <image.h>
```

Inheritance diagram for ecvl::ConstContiguousView< DT >:



## Public Types

- using **basetype** = typename **TypeInfo**< DT >:: **basetype**

## Public Member Functions

- **ConstContiguousView** ( **Image** &img)
- const **basetype** & **operator()** (const std::vector< int > &coords)
- **ConstContiguousIterator**< **basetype** > **Begin** ()
- **ConstContiguousIterator**< **basetype** > **End** ()

## Additional Inherited Members

### 8.2.1 Detailed Description

```
template<DataType DT>
class ecvl::ConstContiguousView< DT >
```

Definition at line 474 of file image.h.

### 8.2.2 Member Typedef Documentation

#### 8.2.2.1 **basetype**

```
template<DataType DT>
using ecvl::ConstContiguousView< DT >:: basetype = typename TypeInfo<DT>:: basetype
```

Definition at line 476 of file image.h.

### 8.2.3 Constructor & Destructor Documentation

#### 8.2.3.1 **ConstContiguousView()**

```
template<DataType DT>
ecvl::ConstContiguousView< DT >:: ConstContiguousView (
    Image & img ) [inline]
```

Definition at line 478 of file image.h.



## 8.2.4 Member Function Documentation

### 8.2.4.1 `Begin()`

```
template<DataType DT>
ConstContiguousIterator< basetype> ecvl::ConstContiguousView< DT >::Begin ( ) [inline]
```

Definition at line 496 of file `image.h`.

### 8.2.4.2 `End()`

```
template<DataType DT>
ConstContiguousIterator< basetype> ecvl::ConstContiguousView< DT >::End ( ) [inline]
```

Definition at line 497 of file `image.h`.

### 8.2.4.3 `operator()`

```
template<DataType DT>
const basetype& ecvl::ConstContiguousView< DT >::operator() (
    const std::vector< int > & coords ) [inline]
```

Definition at line 492 of file `image.h`.

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

- `image.h`

## 8.3 `ecvl::ConstIterator< T >` Struct Template Reference

```
#include <iterators.h>
```

### Public Types

- typedef **ConstIterator** &(ConstIterator::\* **IncrementMemFn**) ()

## Public Member Functions

- **ConstIterator** (const **Image** &img, std::vector< int > pos={})
- **ConstIterator** & **operator++** ()
- const T & **operator\*** () const
- const T \* **operator->** () const
- bool **operator==** (const **ConstIterator** &rhs) const
- bool **operator!=** (const **ConstIterator** &rhs) const

## Public Attributes

- std::vector< int > **pos\_**
- const uint8\_t \* **ptr\_**
- const **Image** \* **img\_**
- **IncrementMemFn** **incrementor** = & **ConstIterator**<T>::IncrementPos

### 8.3.1 Detailed Description

```
template<typename T>
struct ecvl::ConstIterator< T >
```

Definition at line 32 of file iterators.h.

### 8.3.2 Member Typedef Documentation

#### 8.3.2.1 IncrementMemFn

```
template<typename T >
typedef ConstIterator& (ConstIterator::* ecvl::ConstIterator< T >::IncrementMemFn) ()
```

Definition at line 37 of file iterators.h.

### 8.3.3 Constructor & Destructor Documentation

#### 8.3.3.1 ConstIterator()

```
template<typename T >
ConstIterator::ConstIterator (
    const Image & img,
    std::vector< int > pos = {} )
```

Definition at line 30 of file image.h.

### 8.3.4 Member Function Documentation

#### 8.3.4.1 `operator*()`

```
template<typename T >
const T& ecvl::ConstIterator< T >::operator * ( ) const [inline]
```

Definition at line 42 of file `iterators.h`.

#### 8.3.4.2 `operator!=()`

```
template<typename T >
bool ecvl::ConstIterator< T >::operator!= (
    const ConstIterator< T > & rhs ) const [inline]
```

Definition at line 45 of file `iterators.h`.

#### 8.3.4.3 `operator++()`

```
template<typename T >
ConstIterator& ecvl::ConstIterator< T >::operator++ ( ) [inline]
```

Definition at line 41 of file `iterators.h`.

#### 8.3.4.4 `operator->()`

```
template<typename T >
const T* ecvl::ConstIterator< T >::operator-> ( ) const [inline]
```

Definition at line 43 of file `iterators.h`.

#### 8.3.4.5 `operator==()`

```
template<typename T >
bool ecvl::ConstIterator< T >::operator== (
    const ConstIterator< T > & rhs ) const [inline]
```

Definition at line 44 of file `iterators.h`.

### 8.3.5 Member Data Documentation

#### 8.3.5.1 img\_

```
template<typename T >
const Image* ecvl::ConstIterator< T >::img_
```

Definition at line 35 of file iterators.h.

#### 8.3.5.2 incrementor

```
template<typename T >
IncrementMemFn ecvl::ConstIterator< T >::incrementor = & ConstIterator<T>::IncrementPos
```

Definition at line 38 of file iterators.h.

#### 8.3.5.3 pos\_

```
template<typename T >
std::vector<int> ecvl::ConstIterator< T >::pos_
```

Definition at line 33 of file iterators.h.

#### 8.3.5.4 ptr\_

```
template<typename T >
const uint8_t* ecvl::ConstIterator< T >::ptr_
```

Definition at line 34 of file iterators.h.

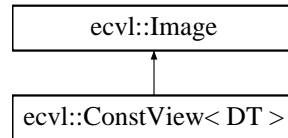
The documentation for this struct was generated from the following files:

- **iterators.h**
- **image.h**
- **iterators\_impl.inc.h**

## 8.4 `ecvl::ConstView< DT >` Class Template Reference

```
#include <image.h>
```

Inheritance diagram for `ecvl::ConstView< DT >`:



### Public Types

- using **basetype** = typename **TypeInfo**< DT >:: **basetype**

### Public Member Functions

- **ConstView** (const **Image** &img)
- const **basetype** & **operator()** (const std::vector< int > &coords)
- **ConstIterator**< **basetype** > **Begin** ()
- **ConstIterator**< **basetype** > **End** ()

### Additional Inherited Members

#### 8.4.1 Detailed Description

```
template<DataType DT>
class ecvl::ConstView< DT >
```

Definition at line 420 of file image.h.

#### 8.4.2 Member Typedef Documentation

##### 8.4.2.1 **basetype**

```
template<DataType DT>
using ecvl::ConstView< DT >:: basetype = typename TypeInfo<DT>:: basetype
```

Definition at line 422 of file image.h.

#### 8.4.3 Constructor & Destructor Documentation

#### 8.4.3.1 ConstView()

```
template<DataType DT>
ecv1::ConstView< DT >:: ConstView (
    const Image & img ) [inline]
```

Definition at line 424 of file image.h.

### 8.4.4 Member Function Documentation

#### 8.4.4.1 Begin()

```
template<DataType DT>
ConstIterator< basetype> ecv1::ConstView< DT >::Begin ( ) [inline]
```

Definition at line 442 of file image.h.

#### 8.4.4.2 End()

```
template<DataType DT>
ConstIterator< basetype> ecv1::ConstView< DT >::End ( ) [inline]
```

Definition at line 443 of file image.h.

#### 8.4.4.3 operator>()

```
template<DataType DT>
const basetype& ecv1::ConstView< DT >::operator() (
    const std::vector< int > & coords ) [inline]
```

Definition at line 438 of file image.h.

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

- **image.h**

## 8.5 **ecv1::ContiguousIterator**< T > Struct Template Reference

```
#include <iterators.h>
```

## Public Member Functions

- **ContiguousIterator** ( **Image** &img, std::vector< int > pos={})
- **ContiguousIterator** & **operator++** ()
- T & **operator\*** () const
- T \* **operator->** () const
- bool **operator==** (const **ContiguousIterator** &rhs) const
- bool **operator!=** (const **ContiguousIterator** &rhs) const

## Public Attributes

- uint8\_t \* **ptr\_**
- **Image** \* **img\_**

### 8.5.1 Detailed Description

```
template<typename T>
struct ecvl::ContiguousIterator< T >
```

Definition at line 53 of file iterators.h.

### 8.5.2 Constructor & Destructor Documentation

#### 8.5.2.1 ContiguousIterator()

```
template<typename T >
ContiguousIterator::ContiguousIterator (
    Image & img,
    std::vector< int > pos = {} )
```

Definition at line 56 of file image.h.

### 8.5.3 Member Function Documentation

#### 8.5.3.1 operator\*()

```
template<typename T >
T& ecvl::ContiguousIterator< T >::operator * ( ) const [inline]
```

Definition at line 59 of file iterators.h.

### 8.5.3.2 operator!=()

```
template<typename T >
bool ecv1::ContiguousIterator< T >::operator!= (
    const ContiguousIterator< T > & rhs ) const [inline]
```

Definition at line 62 of file iterators.h.

### 8.5.3.3 operator++()

```
template<typename T >
ContiguousIterator& ecv1::ContiguousIterator< T >::operator++ ( ) [inline]
```

Definition at line 58 of file iterators.h.

### 8.5.3.4 operator->()

```
template<typename T >
T* ecv1::ContiguousIterator< T >::operator-> ( ) const [inline]
```

Definition at line 60 of file iterators.h.

### 8.5.3.5 operator==()

```
template<typename T >
bool ecv1::ContiguousIterator< T >::operator== (
    const ContiguousIterator< T > & rhs ) const [inline]
```

Definition at line 61 of file iterators.h.

## 8.5.4 Member Data Documentation

### 8.5.4.1 img\_

```
template<typename T >
Image* ecv1::ContiguousIterator< T >::img_
```

Definition at line 55 of file iterators.h.



## 8.5.4.2 ptr\_

```
template<typename T >
uint8_t* ecvl::ContiguousIterator< T >::ptr_
```

Definition at line 54 of file iterators.h.

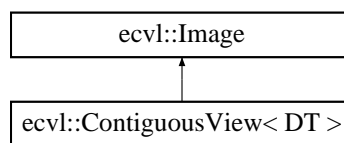
The documentation for this struct was generated from the following files:

- iterators.h
- image.h
- iterators\_impl.inc.h

## 8.6 ecvl::ContiguousView&lt; DT &gt; Class Template Reference

```
#include <image.h>
```

Inheritance diagram for ecvl::ContiguousView< DT >:



## Public Types

- using **basetype** = typename **TypeInfo**< DT >:: **basetype**

## Public Member Functions

- **ContiguousView** ( **Image** &img)
- **basetype** & **operator**() (const std::vector< int > &coords)
- **ContiguousIterator**< **basetype** > **Begin** ()
- **ContiguousIterator**< **basetype** > **End** ()

## Additional Inherited Members

## 8.6.1 Detailed Description

```
template<DataType DT>
class ecvl::ContiguousView< DT >
```

Definition at line 447 of file image.h.

## 8.6.2 Member Typedef Documentation

### 8.6.2.1 basetype

```
template<DataType DT>
using ecv1::ContiguousView< DT >:: basetype = typename TypeInfo<DT>:: basetype
```

Definition at line 449 of file image.h.

## 8.6.3 Constructor & Destructor Documentation

### 8.6.3.1 ContiguousView()

```
template<DataType DT>
ecv1::ContiguousView< DT >:: ContiguousView (
    Image & img ) [inline]
```

Definition at line 451 of file image.h.

## 8.6.4 Member Function Documentation

### 8.6.4.1 Begin()

```
template<DataType DT>
ContiguousIterator< basetype> ecv1::ContiguousView< DT >::Begin ( ) [inline]
```

Definition at line 469 of file image.h.

### 8.6.4.2 End()

```
template<DataType DT>
ContiguousIterator< basetype> ecv1::ContiguousView< DT >::End ( ) [inline]
```

Definition at line 470 of file image.h.

## 8.6.4.3 operator()

```
template<DataType DT>
baseType& ecvl::ContiguousView< DT >::operator() (
    const std::vector< int > & coords ) [inline]
```

Definition at line 465 of file image.h.

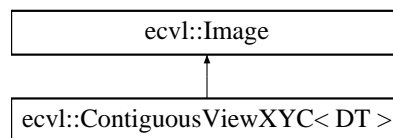
The documentation for this class was generated from the following file:

- **image.h**

## 8.7 ecvl::ContiguousViewXYC&lt; DT &gt; Class Template Reference

```
#include <image.h>
```

Inheritance diagram for ecvl::ContiguousViewXYC< DT >:



## Public Types

- using **baseType** = typename **TypeInfo**< DT >:: **baseType**

## Public Member Functions

- **ContiguousViewXYC** ( **Image** &img)
- int **width** () const
- int **height** () const
- int **channels** () const
- **baseType** & **operator**() (int x, int y, int c)
- **ContiguousIterator**< **baseType** > **Begin** ()
- **ContiguousIterator**< **baseType** > **End** ()

## Additional Inherited Members

## 8.7.1 Detailed Description

```
template<DataType DT>
class ecvl::ContiguousViewXYC< DT >
```

Definition at line 501 of file image.h.

## 8.7.2 Member Typedef Documentation

### 8.7.2.1 basetype

```
template<DataType DT>
using ecv1::ContiguousViewXYC< DT >:: basetype = typename TypeInfo<DT>:: basetype
```

Definition at line 503 of file image.h.

## 8.7.3 Constructor & Destructor Documentation

### 8.7.3.1 ContiguousViewXYC()

```
template<DataType DT>
ecv1::ContiguousViewXYC< DT >:: ContiguousViewXYC (
    Image & img ) [inline]
```

Definition at line 505 of file image.h.

## 8.7.4 Member Function Documentation

### 8.7.4.1 Begin()

```
template<DataType DT>
ContiguousIterator< basetype> ecv1::ContiguousViewXYC< DT >::Begin ( ) [inline]
```

Definition at line 531 of file image.h.

### 8.7.4.2 channels()

```
template<DataType DT>
int ecv1::ContiguousViewXYC< DT >::channels ( ) const [inline]
```

Definition at line 525 of file image.h.

#### 8.7.4.3 End()

```
template<DataType DT>
ContiguousIterator< basetype> ecv1::ContiguousViewXYC< DT >::End ( ) [inline]
```

Definition at line 532 of file image.h.

#### 8.7.4.4 height()

```
template<DataType DT>
int ecv1::ContiguousViewXYC< DT >::height ( ) const [inline]
```

Definition at line 524 of file image.h.

#### 8.7.4.5 operator()

```
template<DataType DT>
basetype& ecv1::ContiguousViewXYC< DT >::operator() (
    int x,
    int y,
    int c ) [inline]
```

Definition at line 527 of file image.h.

#### 8.7.4.6 width()

```
template<DataType DT>
int ecv1::ContiguousViewXYC< DT >::width ( ) const [inline]
```

Definition at line 523 of file image.h.

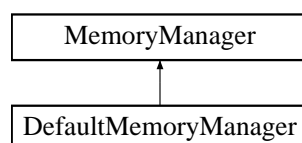
The documentation for this class was generated from the following file:

- **image.h**

## 8.8 DefaultMemoryManager Class Reference

```
#include <memorymanager.h>
```

Inheritance diagram for DefaultMemoryManager:



## Public Member Functions

- virtual uint8\_t\* **Allocate** (size\_t nbytes) override
- virtual void **Deallocate** (uint8\_t \*data) override
- virtual uint8\_t\* **AllocateAndCopy** (size\_t nbytes, uint8\_t \*src) override

## Static Public Member Functions

- static **DefaultMemoryManager** \* **GetInstance** ()

### 8.8.1 Detailed Description

Definition at line 16 of file memorymanager.h.

### 8.8.2 Member Function Documentation

#### 8.8.2.1 Allocate()

```
virtual uint8_t* DefaultMemoryManager::Allocate (  
    size_t nbytes ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 74).

Definition at line 18 of file memorymanager.h.

#### 8.8.2.2 AllocateAndCopy()

```
virtual uint8_t* DefaultMemoryManager::AllocateAndCopy (  
    size_t nbytes,  
    uint8_t * src ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 74).

Definition at line 24 of file memorymanager.h.

#### 8.8.2.3 Deallocate()

```
virtual void DefaultMemoryManager::Deallocate (  
    uint8_t * data ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 75).

Definition at line 21 of file memorymanager.h.

## 8.8.2.4 GetInstance()

```
DefaultMemoryManager * DefaultMemoryManager::GetInstance ( ) [static]
```

Definition at line 3 of file memorymanager.cpp.

The documentation for this class was generated from the following files:

- memorymanager.h
- memorymanager.cpp

## 8.9 ecvl::DivImpl&lt; ST1, ST2, ET &gt; Struct Template Reference

```
#include <arithmetic.h>
```

## Static Public Member Functions

- static constexpr **Image** & \_ (const ST1 &src1, const ST2 &src2, **Image** &dst, bool saturate, ET epsilon)

## 8.9.1 Detailed Description

```
template<typename ST1, typename ST2, typename ET>
struct ecvl::DivImpl< ST1, ST2, ET >
```

Definition at line 476 of file arithmetic.h.

## 8.9.2 Member Function Documentation

## 8.9.2.1 \_()

```
template<typename ST1 , typename ST2 , typename ET >
static constexpr Image& ecvl::DivImpl< ST1, ST2, ET >::_ (
    const ST1 & src1,
    const ST2 & src2,
    Image & dst,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 477 of file arithmetic.h.

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

- arithmetic.h

## 8.10 `ecvl::DivImpl< Image, Image, ET >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static constexpr **Image** & `_` (const **Image** &src1, const **Image** &src2, **Image** &dst, bool saturate, ET epsilon)

### 8.10.1 Detailed Description

```
template<typename ET>
struct ecvl::DivImpl< Image, Image, ET >
```

Definition at line 506 of file arithmetic.h.

### 8.10.2 Member Function Documentation

#### 8.10.2.1 `_()`

```
template<typename ET >
static constexpr Image& ecvl::DivImpl< Image, Image, ET >::_ (
    const Image & src1,
    const Image & src2,
    Image & dst,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 507 of file arithmetic.h.

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

- `arithmetic.h`

## 8.11 `ecvl::DivImpl< Image, T, ET >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static constexpr **Image** & `_` (const **Image** &src1, const T &src2, **Image** &dst, bool saturate, ET epsilon)



### 8.11.1 Detailed Description

```
template<typename T, typename ET>
struct ecvl::DivImpl< Image, T, ET >
```

Definition at line 486 of file arithmetic.h.

### 8.11.2 Member Function Documentation

#### 8.11.2.1 \_()

```
template<typename T , typename ET >
static constexpr Image& ecvl::DivImpl< Image, T, ET >::_ (
    const Image & src1,
    const T & src2,
    Image & dst,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 487 of file arithmetic.h.

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

- arithmetic.h

## 8.12 ecvl::DivImpl< T, Image, ET > Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static constexpr Image & \_ (const T &src1, const Image &src2, Image &dst, bool saturate, ET epsilon)

### 8.12.1 Detailed Description

```
template<typename T, typename ET>
struct ecvl::DivImpl< T, Image, ET >
```

Definition at line 496 of file arithmetic.h.

### 8.12.2 Member Function Documentation

8.12.2.1 `_()`

```
template<typename T , typename ET >
static constexpr Image& ecvl::DivImpl< T, Image, ET >::_ (
    const T & src1,
    const Image & src2,
    Image & dst,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 497 of file arithmetic.h.

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

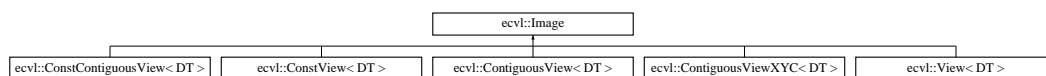
- arithmetic.h

8.13 **ecvl::Image** Class Reference

**Image** (p. 60) class.

```
#include <image.h>
```

Inheritance diagram for **ecvl::Image**:



## Public Member Functions

- template<typename T >  
**Iterator**< T > **Begin** ()  
*Generic non-const Begin Iterator* (p. 71).
- template<typename T >  
**Iterator**< T > **End** ()  
*Generic non-const End Iterator* (p. 71).
- template<typename T >  
**ConstIterator**< T > **Begin** () const  
*Generic const Begin Iterator* (p. 71).
- template<typename T >  
**ConstIterator**< T > **End** () const  
*Generic const End Iterator* (p. 71).
- template<typename T >  
**ContiguousIterator**< T > **ContiguousBegin** ()  
*Contiguous non-const Begin Iterator* (p. 71).
- template<typename T >  
**ContiguousIterator**< T > **ContiguousEnd** ()  
*Contiguous non-const End Iterator* (p. 71).
- template<typename T >  
**ConstContiguousIterator**< T > **ContiguousBegin** () const  
*Contiguous const Begin Iterator* (p. 71).

- `template<typename T >`  
**ConstContiguousIterator**< T > **ContiguousEnd** () const  
*Contiguous const End Iterator* (p. 71).
- **Image** ()  
*Default constructor.*
- **Image** (const std::vector< int > &dims, **DataType** elemtype, std::string channels, **ColorType** colortype)  
*Initializing constructor.*
- **Image** (const **Image** &img)  
*Copy constructor.*
- **Image** ( **Image** &&img)  
*Move constructor.*
- **Image** & **operator=** ( **Image** rhs)
- void **Create** (const std::vector< int > &dims, **DataType** elemtype, std::string channels, **ColorType** colortype)  
*Allocates new contiguous data if needed.*
- **~Image** ()  
*Destructor.*
- bool **IsEmpty** () const  
*To check whether the **Image** (p. 60) contains or not data, regardless the owning status.*
- bool **IsOwner** () const  
*To check whether the **Image** (p. 60) is owner of the data.*
- uint8\_t \* **Ptr** (const std::vector< int > &coords)  
*Returns a non-const pointer to data at given coordinates.*
- const uint8\_t \* **Ptr** (const std::vector< int > &coords) const  
*Returns a const pointer to data at given coordinates.*

## Public Attributes

- **DataType** elemtype\_  
*Type of **Image** (p. 60) pixels, must be one of the values available in **DataType** (p. 17).*
- uint8\_t elemsize\_  
*Size (in bytes) of **Image** (p. 60) pixels.*
- std::vector< int > dims\_  
*Vector of **Image** (p. 60) dimensions. Each dimension is given in pixels/voxels.*
- std::vector< int > strides\_  
*Vector of **Image** (p. 60) strides.*
- std::string channels\_  
*String which describes how **Image** (p. 60) planes are organized.*
- **ColorType** colortype\_  
***Image** (p. 60) ColorType.*
- uint8\_t \* data\_  
*Pointer to **Image** (p. 60) data.*
- size\_t datasize\_  
*Size of **Image** (p. 60) data in bytes.*
- bool contiguous\_  
*Whether the image is stored contiguously or not in memory.*
- **MetaData** \* meta\_  
*Pointer to **Image** (p. 60) **MetaData** (p. 75).*
- **MemoryManager** \* mem\_  
*Pointer to the **MemoryManager** (p. 74) employed by the **Image** (p. 60).*

## Friends

- void **swap** ( **Image** &lhs, **Image** &rhs)

### 8.13.1 Detailed Description

**Image** (p. 60) class.

Definition at line 39 of file image.h.

### 8.13.2 Constructor & Destructor Documentation

#### 8.13.2.1 **Image()** [1/4]

```
ecvl::Image::Image ( ) [inline]
```

Default constructor.

The default constructor creates an empty image without any data.

Definition at line 172 of file image.h.

#### 8.13.2.2 **Image()** [2/4]

```
ecvl::Image::Image (
    const std::vector< int > & dims,
    DataType elemtype,
    std::string channels,
    ColorType colortype ) [inline]
```

Initializing constructor.

The initializing constructor creates a proper image and allocates the data.

Definition at line 191 of file image.h.

#### 8.13.2.3 **Image()** [3/4]

```
ecvl::Image::Image (
    const Image & img ) [inline]
```

Copy constructor.

The copy constructor creates an new **Image** (p. 60) copying (Deep Copy) the input one. The new **Image** (p. 60) will be contiguous regardless of the contiguity of the to be copied **Image** (p. 60).

Definition at line 222 of file image.h.

## 8.13.2.4 Image() [4/4]

```
ecvl::Image::Image (
    Image && img ) [inline]
```

Move constructor.

Move constructor

Definition at line 272 of file image.h.

## 8.13.2.5 ~Image()

```
ecvl::Image::~~Image ( ) [inline]
```

Destructor.

If the **Image** (p. 60) is the owner of data they will be deallocate. Otherwise nothing will happen.

Definition at line 327 of file image.h.

## 8.13.3 Member Function Documentation

## 8.13.3.1 Begin() [1/2]

```
template<typename T >
Iterator<T> ecvl::Image::Begin ( ) [inline]
```

Generic non-const Begin **Iterator** (p. 71).

This function gives you a non-const generic Begin **Iterator** (p. 71) that can be used both for contiguous and non-contiguous non-const Images. It is useful to iterate over a non-const **Image** (p. 60). If the **Image** (p. 60) is contiguous prefer the use of **ContiguousIterator** which in most cases improve the performance.

Definition at line 108 of file image.h.

## 8.13.3.2 Begin() [2/2]

```
template<typename T >
ConstIterator<T> ecvl::Image::Begin ( ) const [inline]
```

Generic const Begin **Iterator** (p. 71).

This function gives you a const generic Begin **Iterator** (p. 71) that can be used both for contiguous and non-contiguous const Images. It is useful to iterate over a const **Image** (p. 60). If the **Image** (p. 60) is contiguous prefer the use of **ConstContiguousIterator** (p. 39) which in most cases improve the performance.

Definition at line 125 of file image.h.

### 8.13.3.3 ContiguousBegin() [1/2]

```
template<typename T >
ContiguousIterator<T> ecvl::Image::ContiguousBegin ( ) [inline]
```

Contiguous non-const Begin **Iterator** (p. 71).

This function gives you a contiguous non-const Begin **Iterator** (p. 71) that can be used only for contiguous Images. If the **Image** (p. 60) is contiguous it is preferable to the non-contiguous iterator since it has usually better performance.

Definition at line 142 of file image.h.

### 8.13.3.4 ContiguousBegin() [2/2]

```
template<typename T >
ConstContiguousIterator<T> ecvl::Image::ContiguousBegin ( ) const [inline]
```

Contiguous const Begin **Iterator** (p. 71).

This function gives you a contiguous const Begin **Iterator** (p. 71) that can be used only for contiguous Images. If the **Image** (p. 60) is contiguous it is preferable to the non-contiguous iterator since it has usually better performance.

Definition at line 159 of file image.h.

### 8.13.3.5 ContiguousEnd() [1/2]

```
template<typename T >
ContiguousIterator<T> ecvl::Image::ContiguousEnd ( ) [inline]
```

Contiguous non-const End **Iterator** (p. 71).

This function gives you a contiguous non-const End **Iterator** (p. 71) that can be used only for contiguous Images.

Definition at line 150 of file image.h.

### 8.13.3.6 ContiguousEnd() [2/2]

```
template<typename T >
ConstContiguousIterator<T> ecvl::Image::ContiguousEnd ( ) const [inline]
```

Contiguous const End **Iterator** (p. 71).

This function gives you a contiguous const End **Iterator** (p. 71) that can be used only for contiguous Images.

Definition at line 166 of file image.h.

## 8.13.3.7 Create()

```
void ecvl::Image::Create (
    const std::vector< int > & dims,
    DataType elemtype,
    std::string channels,
    ColorType colortype )
```

Allocates new contiguous data if needed.

The Create method allocates **Image** (p. 60) data as specified by the input parameters. The procedure tries to avoid the allocation of new memory when possible. The resulting image will be contiguous in any case. Calling this method on an **Image** (p. 60) that does not own data will always cause a new allocation, and the **Image** (p. 60) will become the owner of the data.

## Parameters

## Parameters

in	<i>dims</i>	New <b>Image</b> (p. 60) dimensions.
in	<i>elemtype</i>	New <b>Image</b> (p. 60) <b>DataType</b> .
in	<i>channels</i>	New <b>Image</b> (p. 60) channels.
in	<i>colortype</i>	New <b>Image</b> (p. 60) colortype.

Definition at line 8 of file image.cpp.

## 8.13.3.8 End() [1/2]

```
template<typename T >
Iterator<T> ecvl::Image::End ( ) [inline]
```

Generic non-const End **Iterator** (p. 71).

This function gives you a non-const generic End **Iterator** (p. 71) that can be used both for contiguous and non-contiguous non-const Images. It is useful to iterate over over a non-const **Image** (p. 60).

Definition at line 116 of file image.h.

## 8.13.3.9 End() [2/2]

```
template<typename T >
ConstIterator<T> ecvl::Image::End ( ) const [inline]
```

Generic const End **Iterator** (p. 71).

This function gives you a const generic End **Iterator** (p. 71) that can be used both for contiguous and non-contiguous const Images. It is useful to iterate over a const **Image** (p. 60).

Definition at line 133 of file image.h.

#### 8.13.3.10 IsEmpty()

```
bool ecvl::Image::IsEmpty ( ) const [inline]
```

To check whether the **Image** (p. 60) contains or not data, regardless the owning status.

Definition at line 333 of file image.h.

#### 8.13.3.11 IsOwner()

```
bool ecvl::Image::IsOwner ( ) const [inline]
```

To check whether the **Image** (p. 60) is owner of the data.

Definition at line 336 of file image.h.

#### 8.13.3.12 operator=()

```
Image& ecvl::Image::operator= (
    Image rhs ) [inline]
```

Definition at line 303 of file image.h.

#### 8.13.3.13 Ptr() [1/2]

```
uint8_t* ecvl::Image::Ptr (
    const std::vector< int > & coords ) [inline]
```

Returns a non-const pointer to data at given coordinates.

Definition at line 339 of file image.h.

#### 8.13.3.14 Ptr() [2/2]

```
const uint8_t* ecvl::Image::Ptr (
    const std::vector< int > & coords ) const [inline]
```

Returns a const pointer to data at given coordinates.

Definition at line 344 of file image.h.



### 8.13.4 Friends And Related Function Documentation

#### 8.13.4.1 swap

```
void swap (
    Image & lhs,
    Image & rhs ) [friend]
```

Definition at line 288 of file image.h.

### 8.13.5 Member Data Documentation

#### 8.13.5.1 channels\_

```
std::string ecvl::Image::channels_
```

String which describes how **Image** (p. 60) planes are organized.

A single character provides the information related to the corresponding channel. The possible values are:

- 'x': horizontal spatial dimension
- 'y': vertical spatial dimension
- 'z': depth spatial dimension
- 'c': color dimension
- 't': temporal dimension
- 'o': any other dimension For example, "xyc" describes a 2-dimensional **Image** (p.60) structured in color planes. This could be for example a **ColorType::GRAY** (p. 16) **Image** (p. 60) with `dims_[2] = 1` or a **ColorType::RGB** (p. 16) **Image** (p. 60) with `dims_[2] = 3` and so on. The **ColorType** constrains the value of the dimension corresponding to the color channel. Another example is "cxy" with `dims_[0] = 3` and **ColorType::BGR** (p. 16). In this case the color dimension is the one which changes faster as it is done in other libraries such as OpenCV.

Definition at line 52 of file image.h.

#### 8.13.5.2 colortype\_

```
ColorType ecvl::Image::colortype_
```

**Image** (p. 60) **ColorType**.

If this is different from **ColorType::none** (p. 17) the `channels_` string must contain a 'c' and the corresponding dimension must have the appropriate value. See **ColorType** (p. 16) for the possible values.

Definition at line 75 of file image.h.

#### 8.13.5.3 contiguous\_

```
bool ecvl::Image::contiguous_
```

Whether the image is stored contiguously or not in memory.

Definition at line 89 of file image.h.

#### 8.13.5.4 data\_

```
uint8_t* ecvl::Image::data_
```

Pointer to **Image** (p. 60) data.

If the **Image** (p. 60) is not the owner of data, for example when using **Image** (p. 60) views, this attribute will point to the data of another **Image** (p. 60). The possession or not of the data depends on the **MemoryManager** (p. 74).

Definition at line 81 of file image.h.

#### 8.13.5.5 datasize\_

```
size_t ecvl::Image::datasize_
```

Size of **Image** (p. 60) data in bytes.

Definition at line 88 of file image.h.

#### 8.13.5.6 dims\_

```
std::vector<int> ecvl::Image::dims_
```

Vector of **Image** (p. 60) dimensions. Each dimension is given in pixels/voxels.

Definition at line 44 of file image.h.

#### 8.13.5.7 elemsize\_

```
uint8_t ecvl::Image::elemsize_
```

Size (in bytes) of **Image** (p. 60) pixels.

Definition at line 43 of file image.h.

## 8.13.5.8 elemtype\_

**DataType** ecvl::Image::elemtype\_

Type of **Image** (p. 60) pixels, must be one of the values available in **DataType** (p. 17).

Definition at line 41 of file image.h.

## 8.13.5.9 mem\_

**MemoryManager\*** ecvl::Image::mem\_

Pointer to the **MemoryManager** (p. 74) employed by the **Image** (p. 60).

It can be **DefaultMemoryManager** (p. 55) or **ShallowMemoryManager** (p. 78). The former is responsible for allocating and deallocating data, when using the **DefaultMemoryManager** (p. 55) the **Image** (p. 60) is the owner of data. When **ShallowMemoryManager** (p. 78) is employed the **Image** (p. 60) does not own data and operations on memory are not allowed or does not produce any effect.

Definition at line 92 of file image.h.

## 8.13.5.10 meta\_

**MetaData\*** ecvl::Image::meta\_

Pointer to **Image** (p. 60) **MetaData** (p. 75).

Definition at line 91 of file image.h.

## 8.13.5.11 strides\_

`std::vector<int> ecvl::Image::strides_`

Vector of **Image** (p. 60) strides.

Strides represent  
the number of bytes the pointer on data  
has to move to reach the next pixel/voxel  
on the correspondent size.

Definition at line 46 of file image.h.

The documentation for this class was generated from the following files:

- **image.h**
- **image.cpp**

## 8.14 `ecvl::Table1D<_StructFun, Args>::integer<i>` Struct Template Reference

```
#include <datatype_matrix.h>
```

### 8.14.1 Detailed Description

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
template<int i>
struct ecvl::Table1D<_StructFun, Args>::integer<i>
```

Definition at line 15 of file `datatype_matrix.h`.

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

- `datatype_matrix.h`

## 8.15 `ecvl::SignedTable1D<_StructFun, Args>::integer<i>` Struct Template Reference

```
#include <datatype_matrix.h>
```

### 8.15.1 Detailed Description

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
template<int i>
struct ecvl::SignedTable1D<_StructFun, Args>::integer<i>
```

Definition at line 44 of file `datatype_matrix.h`.

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

- `datatype_matrix.h`

## 8.16 `ecvl::Table2D<_StructFun, Args>::integer<i>` Struct Template Reference

```
#include <datatype_matrix.h>
```

### 8.16.1 Detailed Description

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
template<int i>
struct ecvl::Table2D<_StructFun, Args>::integer<i>
```

Definition at line 73 of file `datatype_matrix.h`.

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

- `datatype_matrix.h`

## 8.17 ecvl::Iterator< T > Struct Template Reference

```
#include <iterators.h>
```

### Public Types

- typedef **Iterator** &(Iterator::\* **IncrementMemFn**) ()

### Public Member Functions

- **Iterator** ( **Image** &img, std::vector< int > pos={})
- **Iterator** & **operator++** ()
- T & **operator\*** () const
- T \* **operator->** () const
- bool **operator==** (const **Iterator** &rhs) const
- bool **operator!=** (const **Iterator** &rhs) const

### Public Attributes

- std::vector< int > **pos\_**
- uint8\_t \* **ptr\_**
- **Image** \* **img\_**
- **IncrementMemFn** **incrementor** = & **Iterator**<T>::IncrementPos

### 8.17.1 Detailed Description

```
template<typename T>
struct ecvl::Iterator< T >
```

Definition at line 12 of file iterators.h.

### 8.17.2 Member Typedef Documentation

#### 8.17.2.1 IncrementMemFn

```
template<typename T >
typedef Iterator&(Iterator::* ecvl::Iterator< T >::IncrementMemFn) ()
```

Definition at line 17 of file iterators.h.

### 8.17.3 Constructor & Destructor Documentation

### 8.17.3.1 Iterator()

```
template<typename T >
Iterator::Iterator (
    Image & img,
    std::vector< int > pos = {} )
```

Definition at line 4 of file image.h.

## 8.17.4 Member Function Documentation

### 8.17.4.1 operator\*()

```
template<typename T >
T& ecv1::Iterator< T >::operator * ( ) const [inline]
```

Definition at line 22 of file iterators.h.

### 8.17.4.2 operator!=(())

```
template<typename T >
bool ecv1::Iterator< T >::operator!= (
    const Iterator< T > & rhs ) const [inline]
```

Definition at line 25 of file iterators.h.

### 8.17.4.3 operator++()

```
template<typename T >
Iterator& ecv1::Iterator< T >::operator++ ( ) [inline]
```

Definition at line 21 of file iterators.h.

### 8.17.4.4 operator->()

```
template<typename T >
T* ecv1::Iterator< T >::operator-> ( ) const [inline]
```

Definition at line 23 of file iterators.h.

#### 8.17.4.5 `operator==( )`

```
template<typename T >
bool ecvl::Iterator< T >::operator==(
    const Iterator< T > & rhs ) const [inline]
```

Definition at line 24 of file `iterators.h`.

### 8.17.5 Member Data Documentation

#### 8.17.5.1 `img_`

```
template<typename T >
Image* ecvl::Iterator< T >::img_
```

Definition at line 15 of file `iterators.h`.

#### 8.17.5.2 `incrementor`

```
template<typename T >
IncrementMemFn ecvl::Iterator< T >::incrementor = & Iterator<T>::IncrementPos
```

Definition at line 18 of file `iterators.h`.

#### 8.17.5.3 `pos_`

```
template<typename T >
std::vector<int> ecvl::Iterator< T >::pos_
```

Definition at line 13 of file `iterators.h`.

#### 8.17.5.4 `ptr_`

```
template<typename T >
uint8_t* ecvl::Iterator< T >::ptr_
```

Definition at line 14 of file `iterators.h`.

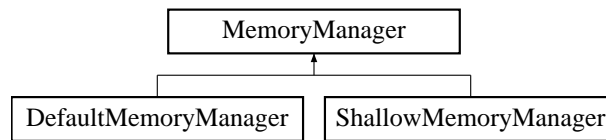
The documentation for this struct was generated from the following files:

- `iterators.h`
- `image.h`
- `iterators_impl.inc.h`

## 8.18 MemoryManager Class Reference

```
#include <memorymanager.h>
```

Inheritance diagram for MemoryManager:



### Public Member Functions

- virtual uint8\_t \* **Allocate** (size\_t nbytes)=0
- virtual void **Deallocate** (uint8\_t \*data)=0
- virtual uint8\_t \* **AllocateAndCopy** (size\_t nbytes, uint8\_t \*src)=0
- virtual ~**MemoryManager** ()

### 8.18.1 Detailed Description

Definition at line 8 of file memorymanager.h.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 ~MemoryManager()

```
virtual MemoryManager::~MemoryManager ( ) [inline], [virtual]
```

Definition at line 13 of file memorymanager.h.

### 8.18.3 Member Function Documentation

#### 8.18.3.1 Allocate()

```
virtual uint8_t* MemoryManager::Allocate (
    size_t nbytes ) [pure virtual]
```

Implemented in **ShallowMemoryManager** (p. 79), and **DefaultMemoryManager** (p. 56).



### 8.18.3.2 AllocateAndCopy()

```
virtual uint8_t* MemoryManager::AllocateAndCopy (
    size_t nbytes,
    uint8_t * src ) [pure virtual]
```

Implemented in **ShallowMemoryManager** (p. 79), and **DefaultMemoryManager** (p. 56).

### 8.18.3.3 Deallocate()

```
virtual void MemoryManager::Deallocate (
    uint8_t * data ) [pure virtual]
```

Implemented in **ShallowMemoryManager** (p. 79), and **DefaultMemoryManager** (p. 56).

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

- **memorymanager.h**

## 8.19 ecvl::MetaData Class Reference

```
#include <image.h>
```

### Public Member Functions

- virtual bool **Query** (const std::string &name, std::string &value) const =0
- virtual **~MetaData** ()

### 8.19.1 Detailed Description

Definition at line 17 of file image.h.

### 8.19.2 Constructor & Destructor Documentation

#### 8.19.2.1 ~MetaData()

```
virtual ecvl::MetaData::~~MetaData ( ) [inline], [virtual]
```

Definition at line 20 of file image.h.

### 8.19.3 Member Function Documentation

#### 8.19.3.1 Query()

```
virtual bool ecvl::MetaData::Query (
    const std::string & name,
    std::string & value ) const [pure virtual]
```

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

- **image.h**

## 8.20 filesystem::path Class Reference

```
#include <filesystem.h>
```

### Public Member Functions

- **path** ()
- **path** (const std::string &p)
- **path** & **operator**/= (const **path** &p)
- **path** & **operator**= (const std::string &s)
- **path** & **operator**= (const **path** &p)
- std::string **string** () const
- **path** **parent\_path** () const
- **path** **stem** () const

#### 8.20.1 Detailed Description

Definition at line 9 of file filesystem.h.

#### 8.20.2 Constructor & Destructor Documentation

##### 8.20.2.1 path() [1/2]

```
filesystem::path::path ( ) [inline]
```

Definition at line 12 of file filesystem.h.

### 8.20.2.2 path() [2/2]

```
filesystem::path::path (  
    const std::string & p ) [inline], [explicit]
```

Definition at line 14 of file filesystem.h.

## 8.20.3 Member Function Documentation

### 8.20.3.1 operator/=()

```
path& filesystem::path::operator/= (  
    const path & p ) [inline]
```

Definition at line 20 of file filesystem.h.

### 8.20.3.2 operator=() [1/2]

```
path& filesystem::path::operator= (  
    const std::string & s ) [inline]
```

Definition at line 48 of file filesystem.h.

### 8.20.3.3 operator=() [2/2]

```
path& filesystem::path::operator= (  
    const path & p ) [inline]
```

Definition at line 55 of file filesystem.h.

### 8.20.3.4 parent\_path()

```
path filesystem::path::parent_path ( ) const [inline]
```

Definition at line 66 of file filesystem.h.

#### 8.20.3.5 stem()

```
path filesystem::path::stem ( ) const [inline]
```

Definition at line 80 of file filesystem.h.

#### 8.20.3.6 string()

```
std::string filesystem::path::string ( ) const [inline]
```

Definition at line 61 of file filesystem.h.

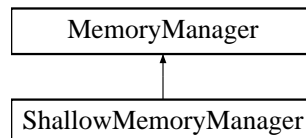
The documentation for this class was generated from the following files:

- **filesystem.h**
- **filesystem.cc**

## 8.21 ShallowMemoryManager Class Reference

```
#include <memorymanager.h>
```

Inheritance diagram for ShallowMemoryManager:



### Public Member Functions

- virtual uint8\_t \* **Allocate** (size\_t nbytes) override
- virtual void **Deallocate** (uint8\_t \*data) override
- virtual uint8\_t \* **AllocateAndCopy** (size\_t nbytes, uint8\_t \*src) override

### Static Public Member Functions

- static **ShallowMemoryManager** \* **GetInstance** ( )

#### 8.21.1 Detailed Description

Definition at line 31 of file memorymanager.h.

## 8.21.2 Member Function Documentation

### 8.21.2.1 Allocate()

```
virtual uint8_t* ShallowMemoryManager::Allocate (
    size_t nbytes ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 74).

Definition at line 33 of file memorymanager.h.

### 8.21.2.2 AllocateAndCopy()

```
virtual uint8_t* ShallowMemoryManager::AllocateAndCopy (
    size_t nbytes,
    uint8_t * src ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 74).

Definition at line 37 of file memorymanager.h.

### 8.21.2.3 Deallocate()

```
virtual void ShallowMemoryManager::Deallocate (
    uint8_t * data ) [inline], [override], [virtual]
```

Implements **MemoryManager** (p. 75).

Definition at line 36 of file memorymanager.h.

### 8.21.2.4 GetInstance()

```
ShallowMemoryManager * ShallowMemoryManager::GetInstance ( ) [static]
```

Definition at line 10 of file memorymanager.cpp.

The documentation for this class was generated from the following files:

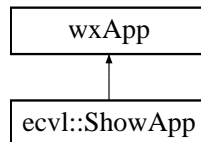
- **memorymanager.h**
- **memorymanager.cpp**

## 8.22 ecvl::ShowApp Class Reference

**ShowApp** (p. 80) is a custom wxApp which allows you to visualize an ECVL **Image** (p. 60).

```
#include <gui.h>
```

Inheritance diagram for ecvl::ShowApp:



### Public Member Functions

- **bool OnInit ()**  
*Initialization function. Starts the main loop of the application.*
- **ShowApp** (const **Image** &img)  
*Constructor.*

#### 8.22.1 Detailed Description

**ShowApp** (p. 80) is a custom wxApp which allows you to visualize an ECVL **Image** (p. 60).

Definition at line 32 of file gui.h.

#### 8.22.2 Constructor & Destructor Documentation

##### 8.22.2.1 ShowApp()

```
ecvl::ShowApp::ShowApp (  
    const Image & img ) [inline]
```

Constructor.

The constructor creates a **ShowApp** (p. 80) initializing its **Image** (p. 60) with the given input **Image** (p. 60).

Definition at line 50 of file gui.h.

#### 8.22.3 Member Function Documentation

8.22.3.1 `OnInit()`

```
bool ecvl::ShowApp::OnInit ( )
```

Initialization function. Starts the main loop of the application.

The **OnInit()** (p. 80) function creates a `wxFrame` which has the width and the height of the **Image** (p. 60) that has to be shown. It also creates the **wxImagePanel** (p. 107) which contains the frame and employs the conversion from **Image** (p. 60) to `wxImage`. It set the `wxImage` in the frame and starts the main loop of the **ShowApp** (p. 80).

Definition at line 72 of file `gui.cpp`.

The documentation for this class was generated from the following files:

- `gui.h`
- `gui.cpp`

8.23 `ecvl::SignedTable1D<_StructFun, Args>` Struct Template Reference

```
#include <datatype_matrix.h>
```

## Classes

- struct `integer`

## Public Types

- using `fun_type` = `decltype(&_StructFun< static_cast< DataType>(0), Args...>::)`

## Public Member Functions

- `template<int i>`  
`constexpr void FillData ( integer< i> )`
- `constexpr void FillData ( integer< DataTypeSignedSize()> )`
- `constexpr SignedTable1D ()`
- `fun_type operator() ( DataType dt) const`

## Public Attributes

- `fun_type data [ DataTypeSignedSize() ]`

## 8.23.1 Detailed Description

```
template<template< DataType, typename ...>class _StructFun, typename ... Args>
struct ecvl::SignedTable1D<_StructFun, Args>
```

Definition at line 39 of file `datatype_matrix.h`.

## 8.23.2 Member Typedef Documentation

### 8.23.2.1 fun\_type

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
using ecv1::SignedTable1D< _StructFun, Args >:: fun_type = decltype(&_StructFun<static_↵
cast< DataType>(0), Args...>::_)
```

Definition at line 41 of file datatype\_matrix.h.

## 8.23.3 Constructor & Destructor Documentation

### 8.23.3.1 SignedTable1D()

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
constexpr ecv1::SignedTable1D< _StructFun, Args >:: SignedTable1D ( ) [inline]
```

Definition at line 55 of file datatype\_matrix.h.

## 8.23.4 Member Function Documentation

### 8.23.4.1 FillData() [1/2]

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
template<int i>
constexpr void ecv1::SignedTable1D< _StructFun, Args >::FillData (
    integer< i > ) [inline]
```

Definition at line 47 of file datatype\_matrix.h.

### 8.23.4.2 FillData() [2/2]

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
constexpr void ecv1::SignedTable1D< _StructFun, Args >::FillData (
    integer< DataTypeSignedSize()> ) [inline]
```

Definition at line 53 of file datatype\_matrix.h.



8.23.4.3 `operator()`

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::SignedTable1D< _StructFun, Args >::operator() (
    DataType dt ) const [inline]
```

Definition at line 59 of file `datatype_matrix.h`.

## 8.23.5 Member Data Documentation

8.23.5.1 `data`

```
template<template< DataType, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::SignedTable1D< _StructFun, Args >::data[ DataTypeSignedSize() ]
```

Definition at line 63 of file `datatype_matrix.h`.

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

- `datatype_matrix.h`

8.24 `ecvl::StructAdd< a, b >` Struct Template Reference

## Static Public Member Functions

- static void `_ ( Image &src1_dst, const Image &src2)`

## 8.24.1 Detailed Description

```
template<DataType a, DataType b>
struct ecvl::StructAdd< a, b >
```

Definition at line 23 of file `arithmetic.cpp`.

## 8.24.2 Member Function Documentation

### 8.24.2.1 `_()`

```
template<DataType a, DataType b>
static void ecvl::StructAdd< a, b >::_ (
    Image & src1_dst,
    const Image & src2 ) [inline], [static]
```

Definition at line 24 of file arithmetic.cpp.

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

- arithmetic.cpp

## 8.25 **ecvl::StructCopyImage**< SDT, DDT > Struct Template Reference

```
#include <image.h>
```

### Static Public Member Functions

- static void `_` (const **Image** &src, **Image** &dst)

### 8.25.1 Detailed Description

```
template<DataType SDT, DataType DDT>
struct ecvl::StructCopyImage< SDT, DDT >
```

Definition at line 552 of file image.h.

### 8.25.2 Member Function Documentation

#### 8.25.2.1 `_()`

```
template<DataType SDT, DataType DDT>
static void ecvl::StructCopyImage< SDT, DDT >::_ (
    const Image & src,
    Image & dst ) [inline], [static]
```

Definition at line 553 of file image.h.

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

- image.h

## 8.26 `ecvl::StructDiv< a, b, ET >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static void `_ (Image &src1_dst, const Image &src2, bool saturate, ET epsilon)`

#### 8.26.1 Detailed Description

```
template<DataType a, DataType b, typename ET>
struct ecvl::StructDiv< a, b, ET >
```

Definition at line 454 of file `arithmetic.h`.

#### 8.26.2 Member Function Documentation

##### 8.26.2.1 `_()`

```
template<DataType a, DataType b, typename ET >
static void ecvl::StructDiv< a, b, ET >::_ (
    Image & src1_dst,
    const Image & src2,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 455 of file `arithmetic.h`.

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

- `arithmetic.h`

## 8.27 `ecvl::StructMul< a, b >` Struct Template Reference

### Static Public Member Functions

- static void `_ (Image &src1_dst, const Image &src2)`

#### 8.27.1 Detailed Description

```
template<DataType a, DataType b>
struct ecvl::StructMul< a, b >
```

Definition at line 57 of file `arithmetic.cpp`.

## 8.27.2 Member Function Documentation

### 8.27.2.1 \_()

```
template<DataType a, DataType b>
static void ecvl::StructMul< a, b >::_ (
    Image & src1_dst,
    const Image & src2 ) [inline], [static]
```

Definition at line 58 of file arithmetic.cpp.

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

- arithmetic.cpp

## 8.28 **ecvl::StructScalarAdd**< DT, T > Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & \_ ( **Image** &img, T value, bool saturate)

### 8.28.1 Detailed Description

```
template<DataType DT, typename T>
struct ecvl::StructScalarAdd< DT, T >
```

Definition at line 123 of file arithmetic.h.

## 8.28.2 Member Function Documentation

### 8.28.2.1 \_()

```
template<DataType DT, typename T >
static Image& ecvl::StructScalarAdd< DT, T >::_ (
    Image & img,
    T value,
    bool saturate ) [inline], [static]
```

Definition at line 124 of file arithmetic.h.

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

- arithmetic.h

## 8.29 `ecvl::StructScalarDiv< DT, T >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & `_` ( **Image** &img, T value, bool saturate)

#### 8.29.1 Detailed Description

```
template<DataType DT, typename T>
struct ecvl::StructScalarDiv< DT, T >
```

Definition at line 258 of file `arithmetic.h`.

#### 8.29.2 Member Function Documentation

##### 8.29.2.1 `_()`

```
template<DataType DT, typename T >
static Image& ecvl::StructScalarDiv< DT, T >::_ (
    Image & img,
    T value,
    bool saturate ) [inline], [static]
```

Definition at line 259 of file `arithmetic.h`.

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

- `arithmetic.h`

## 8.30 `ecvl::StructScalarDivInv< DT, T, ET >` Struct Template Reference

In-place division between an **Image** (p. 60) and a scalar value, without type promotion.

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & `_` (T value, **Image** &img, bool saturate, ET epsilon)

### 8.30.1 Detailed Description

```
template<DataType DT, typename T, typename ET>
struct ecvl::StructScalarDivInv< DT, T, ET >
```

In-place division between an **Image** (p. 60) and a scalar value, without type promotion.

The **Div()** (p. 22) function divides an input **Image** (p. 60) by a scalar value and stores the result in the same image. The type of the image will not change. By default a saturation will be applied. If it is not the desired behavior change the "saturate" parameter to false.

#### Parameters

##### Parameters

<i>in, out</i>	<i>img</i>	<b>Image</b> (p. 60) to be divided (in-place) by a scalar value.
<i>in</i>	<i>value</i>	Scalar value to use for the division.
<i>in</i>	<i>saturate</i>	Whether to apply saturation or not. Default is true.

#### Returns

Reference to the **Image** (p. 60) containing the result of the division.

Definition at line 301 of file arithmetic.h.

### 8.30.2 Member Function Documentation

#### 8.30.2.1 \_()

```
template<DataType DT, typename T , typename ET >
static Image& ecvl::StructScalarDivInv< DT, T, ET >::_ (
    T value,
    Image & img,
    bool saturate,
    ET epsilon ) [inline], [static]
```

Definition at line 302 of file arithmetic.h.

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

- arithmetic.h

### 8.31 ecvl::StructScalarMul< DT, T > Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & `_` ( **Image** &img, T d, bool saturate)

#### 8.31.1 Detailed Description

```
template<DataType DT, typename T>
struct ecvl::StructScalarMul< DT, T >
```

Definition at line 74 of file arithmetic.h.

#### 8.31.2 Member Function Documentation

##### 8.31.2.1 `_()`

```
template<DataType DT, typename T >
static Image& ecvl::StructScalarMul< DT, T >::_ (
    Image & img,
    T d,
    bool saturate ) [inline], [static]
```

Definition at line 75 of file arithmetic.h.

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

- **arithmetic.h**

## 8.32 `ecvl::StructScalarNeg< DT >` Struct Template Reference

### Static Public Member Functions

- static **Image** & `_` ( **Image** &img)

#### 8.32.1 Detailed Description

```
template<DataType DT>
struct ecvl::StructScalarNeg< DT >
```

Definition at line 79 of file arithmetic.cpp.

#### 8.32.2 Member Function Documentation

### 8.32.2.1 `_()`

```
template<DataType DT>
static Image& ecvl::StructScalarNeg< DT >::_ (
    Image & img ) [inline], [static]
```

Definition at line 80 of file arithmetic.cpp.

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

- arithmetic.cpp

## 8.33 `ecvl::StructScalarSub< DT, T >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & `_ ( Image &img, T value, bool saturate)`

### 8.33.1 Detailed Description

```
template<DataType DT, typename T>
struct ecvl::StructScalarSub< DT, T >
```

Definition at line 172 of file arithmetic.h.

### 8.33.2 Member Function Documentation

#### 8.33.2.1 `_()`

```
template<DataType DT, typename T >
static Image& ecvl::StructScalarSub< DT, T >::_ (
    Image & img,
    T value,
    bool saturate ) [inline], [static]
```

Definition at line 173 of file arithmetic.h.

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

- arithmetic.h



## 8.34 `ecvl::StructScalarSubInv< DT, T >` Struct Template Reference

```
#include <arithmetic.h>
```

### Static Public Member Functions

- static **Image** & `_` (T value, **Image** &img, bool saturate)

#### 8.34.1 Detailed Description

```
template<DataType DT, typename T>
struct ecvl::StructScalarSubInv< DT, T >
```

Definition at line 215 of file `arithmetic.h`.

#### 8.34.2 Member Function Documentation

##### 8.34.2.1 `_()`

```
template<DataType DT, typename T >
static Image& ecvl::StructScalarSubInv< DT, T >::_ (
    T value,
    Image & img,
    bool saturate ) [inline], [static]
```

Definition at line 216 of file `arithmetic.h`.

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

- `arithmetic.h`

## 8.35 `ecvl::StructSub< a, b >` Struct Template Reference

### Static Public Member Functions

- static void `_` ( **Image** &src1\_dst, const **Image** &src2)

#### 8.35.1 Detailed Description

```
template<DataType a, DataType b>
struct ecvl::StructSub< a, b >
```

Definition at line 40 of file `arithmetic.cpp`.

### 8.35.2 Member Function Documentation

#### 8.35.2.1 \_()

```
template<DataType a, DataType b>
static void ecvl::StructSub< a, b >::_ (
    Image & src1_dst,
    const Image & src2 ) [inline], [static]
```

Definition at line 41 of file arithmetic.cpp.

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

- arithmetic.cpp

## 8.36 **ecvl::Table1D<\_StructFun, Args >** Struct Template Reference

```
#include <datatype_matrix.h>
```

### Classes

- struct **integer**

### Public Types

- using **fun\_type** = decltype(&\_StructFun< static\_cast< **DataType** >(0), Args... >::\_)

### Public Member Functions

- template<int i>  
constexpr void **FillData** ( **integer**< i >)
- constexpr void **FillData** ( **integer**< **DataTypeSize**()>)
- constexpr **Table1D** ()
- **fun\_type** **operator**() ( **DataType** dt) const

### Public Attributes

- **fun\_type** **data** [ **DataTypeSize**()]

#### 8.36.1 Detailed Description

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
struct ecvl::Table1D< _StructFun, Args >
```

Definition at line 10 of file datatype\_matrix.h.

## 8.36.2 Member Typedef Documentation

### 8.36.2.1 `fun_type`

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
using ecvl::Table1D< _StructFun, Args >:: fun_type = decltype(&_StructFun<static_cast<
DataType>(0), Args...>::_)
```

Definition at line 12 of file `datatype_matrix.h`.

## 8.36.3 Constructor & Destructor Documentation

### 8.36.3.1 `Table1D()`

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
constexpr ecvl::Table1D< _StructFun, Args >:: Table1D ( ) [inline]
```

Definition at line 26 of file `datatype_matrix.h`.

## 8.36.4 Member Function Documentation

### 8.36.4.1 `FillData()` [1/2]

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
template<int i>
constexpr void ecvl::Table1D< _StructFun, Args >::FillData (
    integer< i > ) [inline]
```

Definition at line 18 of file `datatype_matrix.h`.

### 8.36.4.2 `FillData()` [2/2]

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
constexpr void ecvl::Table1D< _StructFun, Args >::FillData (
    integer< DataTypeSize()> ) [inline]
```

Definition at line 24 of file `datatype_matrix.h`.

### 8.36.4.3 operator()

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::Table1D< _StructFun, Args >::operator() (
    DataType dt ) const [inline]
```

Definition at line 30 of file datatype\_matrix.h.

## 8.36.5 Member Data Documentation

### 8.36.5.1 data

```
template<template< DataType DT, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::Table1D< _StructFun, Args >::data[ DataTypeSize() ]
```

Definition at line 34 of file datatype\_matrix.h.

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

- `datatype_matrix.h`

## 8.37 ecvl::Table2D< \_StructFun, Args > Struct Template Reference

```
#include <datatype_matrix.h>
```

### Classes

- struct `integer`

### Public Types

- using `fun_type` = `decltype(&_StructFun< static_cast< DataType >(0), static_cast< DataType >(0), Args... >::_)`

### Public Member Functions

- `template<int i>`  
`constexpr void FillData ( integer< i > )`
- `constexpr void FillData ( integer< DataTypeSize() * DataTypeSize() > )`
- `constexpr Table2D ()`
- `fun_type operator() ( DataType src, DataType dst) const`

## Public Attributes

- **fun\_type** data [ DataTypeSize() \* DataTypeSize() ]

### 8.37.1 Detailed Description

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
struct ecvl::Table2D<_StructFun, Args>
```

Definition at line 68 of file datatype\_matrix.h.

### 8.37.2 Member Typedef Documentation

#### 8.37.2.1 fun\_type

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
using ecvl::Table2D<_StructFun, Args>:: fun_type = decltype(&_StructFun<static_cast<
DataType>(0), static_cast< DataType>(0), Args...>::_)
```

Definition at line 70 of file datatype\_matrix.h.

### 8.37.3 Constructor & Destructor Documentation

#### 8.37.3.1 Table2D()

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
constexpr ecvl::Table2D<_StructFun, Args>:: Table2D ( ) [inline]
```

Definition at line 86 of file datatype\_matrix.h.

### 8.37.4 Member Function Documentation

#### 8.37.4.1 FillData() [1/2]

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
template<int i>
constexpr void ecvl::Table2D<_StructFun, Args>::FillData (
    integer<i> ) [inline]
```

Definition at line 76 of file datatype\_matrix.h.

#### 8.37.4.2 FillData() [2/2]

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
constexpr void ecvl::Table2D< _StructFun, Args >::FillData (
    integer< DataTypeSize() * DataTypeSize() > ) [inline]
```

Definition at line 84 of file datatype\_matrix.h.

#### 8.37.4.3 operator>()()

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::Table2D< _StructFun, Args >::operator() (
    DataType src,
    DataType dst ) const [inline]
```

Definition at line 90 of file datatype\_matrix.h.

### 8.37.5 Member Data Documentation

#### 8.37.5.1 data

```
template<template< DataType, DataType, typename ... >class _StructFun, typename ... Args>
fun_type ecvl::Table2D< _StructFun, Args >::data[ DataTypeSize() * DataTypeSize()]
```

Definition at line 96 of file datatype\_matrix.h.

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

- **datatype\_matrix.h**

## 8.38 **ecvl::TypeInfo**< **DataType** > Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = void

#### 8.38.1 Detailed Description

```
template<ecvl::DataType>
struct ecvl::TypeInfo< DataType >
```

Definition at line 33 of file datatype.h.

## 8.38.2 Member Typedef Documentation

### 8.38.2.1 `basetype`

```
template<ecvl::DataType >  
using ecvl::TypeInfo< DataType >:: basetype = void
```

Definition at line 33 of file `datatype.h`.

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

- `datatype.h`

## 8.39 `ecvl::TypeInfo< ecvl::DataType::float32 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = float

### 8.39.1 Detailed Description

```
template<>  
struct ecvl::TypeInfo< ecvl::DataType::float32 >
```

Definition at line 6 of file `datatype.h`.

## 8.39.2 Member Typedef Documentation

### 8.39.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::float32 >:: basetype = float
```

Definition at line 6 of file `datatype.h`.

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

- `datatype.h`

## 8.40 `ecvl::TypeInfo< ecvl::DataType::float64 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = double

#### 8.40.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::float64 >
```

Definition at line 7 of file datatype.h.

#### 8.40.2 Member Typedef Documentation

##### 8.40.2.1 **basetype**

```
using ecvl::TypeInfo< ecvl::DataType::float64 >:: basetype = double
```

Definition at line 7 of file datatype.h.

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

- **datatype.h**

## 8.41 `ecvl::TypeInfo< ecvl::DataType::int16 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = int16\_t

#### 8.41.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::int16 >
```

Definition at line 3 of file datatype.h.



### 8.41.2 Member Typedef Documentation

#### 8.41.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::int16 >:: basetype = int16_t
```

Definition at line 3 of file `datatype.h`.

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

- `datatype.h`

## 8.42 `ecvl::TypeInfo< ecvl::DataType::int32 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **`basetype`** = `int32_t`

### 8.42.1 Detailed Description

```
template<>  
struct ecvl::TypeInfo< ecvl::DataType::int32 >
```

Definition at line 4 of file `datatype.h`.

### 8.42.2 Member Typedef Documentation

#### 8.42.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::int32 >:: basetype = int32_t
```

Definition at line 4 of file `datatype.h`.

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

- `datatype.h`

## 8.43 `ecvl::TypeInfo< ecvl::DataType::int64 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = int64\_t

#### 8.43.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::int64 >
```

Definition at line 5 of file datatype.h.

#### 8.43.2 Member Typedef Documentation

##### 8.43.2.1 **basetype**

```
using ecvl::TypeInfo< ecvl::DataType::int64 >:: basetype = int64_t
```

Definition at line 5 of file datatype.h.

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

- **datatype.h**

## 8.44 `ecvl::TypeInfo< ecvl::DataType::int8 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = int8\_t

#### 8.44.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::int8 >
```

Definition at line 2 of file datatype.h.

## 8.44.2 Member Typedef Documentation

### 8.44.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::int8 >:: basetype = int8_t
```

Definition at line 2 of file `datatype.h`.

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

- `datatype.h`

## 8.45 `ecvl::TypeInfo< ecvl::DataType::none >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using `basetype` = void

### 8.45.1 Detailed Description

```
template<>  
struct ecvl::TypeInfo< ecvl::DataType::none >
```

Definition at line 7 of file `datatype.h`.

## 8.45.2 Member Typedef Documentation

### 8.45.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::none >:: basetype = void
```

Definition at line 7 of file `datatype.h`.

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

- `datatype.h`

## 8.46 `ecvl::TypeInfo< ecvl::DataType::uint16 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = uint16\_t

#### 8.46.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::uint16 >
```

Definition at line 3 of file datatype.h.

#### 8.46.2 Member Typedef Documentation

##### 8.46.2.1 **basetype**

```
using ecvl::TypeInfo< ecvl::DataType::uint16 >:: basetype = uint16_t
```

Definition at line 3 of file datatype.h.

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

- **datatype.h**

## 8.47 `ecvl::TypeInfo< ecvl::DataType::uint32 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = uint32\_t

#### 8.47.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::uint32 >
```

Definition at line 4 of file datatype.h.

## 8.47.2 Member Typedef Documentation

### 8.47.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::uint32 >:: basetype = uint32_t
```

Definition at line 4 of file `datatype.h`.

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

- `datatype.h`

## 8.48 `ecvl::TypeInfo< ecvl::DataType::uint64 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **`basetype`** = `uint64_t`

### 8.48.1 Detailed Description

```
template<>  
struct ecvl::TypeInfo< ecvl::DataType::uint64 >
```

Definition at line 5 of file `datatype.h`.

## 8.48.2 Member Typedef Documentation

### 8.48.2.1 `basetype`

```
using ecvl::TypeInfo< ecvl::DataType::uint64 >:: basetype = uint64_t
```

Definition at line 5 of file `datatype.h`.

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

- `datatype.h`

## 8.49 `ecvl::TypeInfo< ecvl::DataType::uint8 >` Struct Template Reference

```
#include <datatype.h>
```

### Public Types

- using **basetype** = uint8\_t

#### 8.49.1 Detailed Description

```
template<>
struct ecvl::TypeInfo< ecvl::DataType::uint8 >
```

Definition at line 2 of file datatype.h.

#### 8.49.2 Member Typedef Documentation

##### 8.49.2.1 **basetype**

```
using ecvl::TypeInfo< ecvl::DataType::uint8 >:: basetype = uint8_t
```

Definition at line 2 of file datatype.h.

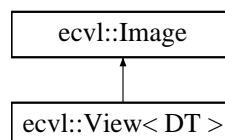
The documentation for this struct was generated from the following file:

- **datatype.h**

## 8.50 `ecvl::View< DT >` Class Template Reference

```
#include <image.h>
```

Inheritance diagram for `ecvl::View< DT >`:



### Public Types

- using **basetype** = typename **TypeInfo< DT >:: basetype**

## Public Member Functions

- **View** ( **Image** &img)
- **View** ( **Image** &img, const std::vector< int > &start, const std::vector< int > &size)
- **basetype** & **operator()** (const std::vector< int > &coords)
- **Iterator**< **basetype** > **Begin** ()
- **Iterator**< **basetype** > **End** ()

## Additional Inherited Members

### 8.50.1 Detailed Description

```
template<DataType DT>
class ecvl::View< DT >
```

Definition at line 353 of file image.h.

### 8.50.2 Member Typedef Documentation

#### 8.50.2.1 **basetype**

```
template<DataType DT>
using ecvl::View< DT >:: basetype = typename TypeInfo<DT>:: basetype
```

Definition at line 355 of file image.h.

### 8.50.3 Constructor & Destructor Documentation

#### 8.50.3.1 **View()** [1/2]

```
template<DataType DT>
ecvl::View< DT >:: View (
    Image & img ) [inline]
```

Definition at line 357 of file image.h.

### 8.50.3.2 View() [2/2]

```
template<DataType DT>
ecvl::View< DT >:: View (
    Image & img,
    const std::vector< int > & start,
    const std::vector< int > & size ) [inline]
```

Definition at line 375 of file image.h.

## 8.50.4 Member Function Documentation

### 8.50.4.1 Begin()

```
template<DataType DT>
Iterator< basetype> ecvl::View< DT >::Begin ( ) [inline]
```

Definition at line 415 of file image.h.

### 8.50.4.2 End()

```
template<DataType DT>
Iterator< basetype> ecvl::View< DT >::End ( ) [inline]
```

Definition at line 416 of file image.h.

### 8.50.4.3 operator>()()

```
template<DataType DT>
basetype& ecvl::View< DT >::operator() (
    const std::vector< int > & coords ) [inline]
```

Definition at line 411 of file image.h.

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

- **image.h**

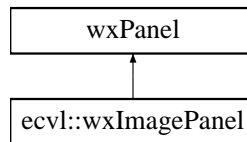


## 8.51 ecvl::wxImagePanel Class Reference

**wxImagePanel** (p. 107) creates a wxPanel to contain an **Image** (p. 60).

```
#include <gui.h>
```

Inheritance diagram for ecvl::wxImagePanel:



### Public Member Functions

- **wxImagePanel** (wxFrame \*parent)
- void **SetImage** (const wxImage &img)

#### 8.51.1 Detailed Description

**wxImagePanel** (p. 107) creates a wxPanel to contain an **Image** (p. 60).

Definition at line 13 of file gui.h.

#### 8.51.2 Constructor & Destructor Documentation

##### 8.51.2.1 wxImagePanel()

```
ecvl::wxImagePanel::wxImagePanel (  
    wxFrame * parent ) [inline]
```

Definition at line 23 of file gui.h.

#### 8.51.3 Member Function Documentation

##### 8.51.3.1 SetImage()

```
void ecvl::wxImagePanel::SetImage (  
    const wxImage & img )
```

Definition at line 10 of file gui.cpp.

The documentation for this class was generated from the following files:

- **gui.h**
- **gui.cpp**



## Chapter 9

# File Documentation

### 9.1 arithmetic.cpp File Reference

```
#include "ecvl/core/arithmetic.h"
```

#### Classes

- struct **ecvl::StructAdd**< **a**, **b** >
- struct **ecvl::StructSub**< **a**, **b** >
- struct **ecvl::StructMul**< **a**, **b** >
- struct **ecvl::StructScalarNeg**< **DT** >

#### Namespaces

- **ecvl**

#### Macros

- #define **STANDARD\_INPLACE\_OPERATION**(Function, TemplatImplementation)
- #define **STANDARD\_NON\_INPLACE\_OPERATION**(Function)

#### Functions

- void **ecvl::Add** (Image &src1\_dst, const Image &src2)
- void **ecvl::Sub** (Image &src1\_dst, const Image &src2)
- void **ecvl::Mul** (Image &src1\_dst, const Image &src2)
- Image & **ecvl::Neg** (Image &img)  
*In-place divion between a scalar value and an **Image** (p. 60), without type promotion.*
- void **ecvl::Mul** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Multiplies two Image(s) and stores the result in a third **Image** (p. 60).*
- void **ecvl::Add** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Adds two Image(s) and stores the result in a third **Image** (p. 60).*
- void **ecvl::Sub** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Subtracts two Image(s) and stores the result in a third **Image** (p. 60).*

## 9.1.1 Macro Definition Documentation

### 9.1.1.1 STANDARD\_INPLACE\_OPERATION

```
#define STANDARD_INPLACE_OPERATION(  
    Function,  
    TemplateImplementation )
```

#### Value:

```
void Function(Image& src1_dst, const Image& src2)      \  
{                                                    \  
    static constexpr Table2D<TemplateImplementation> table; \  
    table(src1_dst.elemtype_, src2.elemtype_)(src1_dst, src2); \  
}
```

Definition at line 14 of file arithmetic.cpp.

### 9.1.1.2 STANDARD\_NON\_INPLACE\_OPERATION

```
#define STANDARD_NON_INPLACE_OPERATION(  
    Function )
```

#### Value:

```
void Function(const Image& src1, const Image& src2, Image& dst, DataType dst_type, bool saturate) \  
{                                                                                               \  
    if (src1.dims_ != src2.dims_ || src1.channels_ != src2.channels_) { \  
        throw std::runtime_error("Source images must have the same dimensions and channels."); \  
    }                                                                                               \  
    if (!dst.IsOwner()) { \  
        if (src1.dims_ != dst.dims_ || src1.channels_ != dst.channels_) { \  
            throw std::runtime_error("Non-owning data destination image must have the \  
                same dimensions and channels as the sources."); \  
        }                                                                                               \  
    }                                                                                               \  
    CopyImage(src1, dst, dst_type); \  
    Function(dst, src2); \  
}
```

Definition at line 108 of file arithmetic.cpp.

## 9.2 arithmetic.h File Reference

```
#include <type_traits>  
#include "ecvl/core/datatype_matrix.h"  
#include "ecvl/core/image.h"  
#include "ecvl/core/standard_errors.h"
```

## Classes

- struct **ecvl::StructScalarMul**< DT, T >
- struct **ecvl::StructScalarAdd**< DT, T >
- struct **ecvl::StructScalarSub**< DT, T >
- struct **ecvl::StructScalarSubInv**< DT, T >
- struct **ecvl::StructScalarDiv**< DT, T >
- struct **ecvl::StructScalarDivInv**< DT, T, ET >  
*In-place division between an **Image** (p. 60) and a scalar value, without type promotion.*
- struct **ecvl::StructDiv**< a, b, ET >
- struct **ecvl::DivImpl**< ST1, ST2, ET >
- struct **ecvl::DivImpl**< Image, T, ET >
- struct **ecvl::DivImpl**< T, Image, ET >
- struct **ecvl::DivImpl**< Image, Image, ET >

## Namespaces

- **ecvl**

## Functions

- template<DataType ODT, typename IDT >  
TypeInfo< ODT >::basetype **ecvl::saturate\_cast** (IDT v)  
*Saturate a value (of any type) to the specified type.*
- template<typename ODT, typename IDT >  
ODT **ecvl::saturate\_cast** (const IDT &v)  
*Saturate a value (of any type) to the specified type.*
- void **ecvl::Add** (Image &src1\_dst, const Image &src2)
- void **ecvl::Sub** (Image &src1\_dst, const Image &src2)
- void **ecvl::Mul** (Image &src1\_dst, const Image &src2)
- template<typename T >  
Image & **ecvl::Mul** (Image &img, T value, bool saturate=true)  
*In-place multiplication between an **Image** (p. 60) and a scalar value, without type promotion.*
- template<typename T >  
Image & **ecvl::Mul** (T value, Image &img, bool saturate=true)
- template<typename T >  
Image & **ecvl::Add** (Image &img, T value, bool saturate=true)  
*In-place addition between an **Image** (p. 60) and a scalar value, without type promotion.*
- template<typename T >  
Image & **ecvl::Add** (T value, Image &img, bool saturate=true)
- template<typename T >  
Image & **ecvl::Sub** (Image &img, T value, bool saturate=true)  
*In-place subtraction between an **Image** (p. 60) and a scalar value, without type promotion.*
- template<typename T >  
Image & **ecvl::Sub** (T value, Image &img, bool saturate=true)  
*In-place subtraction between a scalar value and an **Image** (p. 60), without type promotion.*
- Image & **ecvl::Neg** (Image &img)  
*In-place division between a scalar value and an **Image** (p. 60), without type promotion.*
- void **ecvl::Mul** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Multiplies two Image(s) and stores the result in a third **Image** (p. 60).*
- void **ecvl::Sub** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Subtracts two Image(s) and stores the result in a third **Image** (p. 60).*

- void **ecvl::Add** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true)  
*Adds two Image(s) and stores the result in a third **Image** (p. 60).*
- template<typename ET = double>  
void **ecvl::Div** (const Image &src1, const Image &src2, Image &dst, DataType dst\_type, bool saturate=true, ET epsilon=std::numeric\_limits< double >::min())  
*Divides two Image(s) and stores the result in a third **Image** (p. 60).*
- template<typename ST1, typename ST2, typename ET = double>  
constexpr Image & **ecvl::Div** (const ST1 &src1, const ST2 &src2, Image &dst, bool saturate=true, ET epsilon=std::numeric\_limits< double >::min())

### 9.3 core.cpp File Reference

```
#include "ecvl/core/image.h"
```

### 9.4 core.h File Reference

```
#include "core/arithmetic.h"
#include "core/datatype.h"
#include "core/filesystem.h"
#include "core/image.h"
#include "core/imgcodecs.h"
#include "core/imgproc.h"
#include "core/iterators.h"
#include "core/memorymanager.h"
#include "core/support_opencv.h"
```

### 9.5 datatype.cpp File Reference

```
#include "ecvl/core/datatype.h"
#include "ecvl/core/datatype_tuples.inc.h"
```

#### Namespaces

- **ecvl**

#### Macros

- #define **ECVL\_TUPLE**(name, size, ...) size,

#### Functions

- uint8\_t **ecvl::DataTypeSize** (DataType dt)  
*Provides the size in bytes of a given DataType.*

## 9.5.1 Macro Definition Documentation

### 9.5.1.1 ECVL\_TUPLE

```
#define ECVL_TUPLE(  
    name,  
    size,  
    ... ) size,
```

## 9.6 datatype.h File Reference

```
#include <cstdint>  
#include <limits>  
#include <array>  
#include "datatype_tuples.inc.h"  
#include "datatype_existing_tuples.inc.h"  
#include "datatype_existing_tuples_signed.inc.h"
```

## Classes

- struct **ecvl::TypeInfo**< **DataType** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::int8** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::int16** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::int32** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::int64** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::float32** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::float64** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::uint8** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::uint16** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::uint32** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::uint64** >
- struct **ecvl::TypeInfo**< **ecvl::DataType::none** >

## Namespaces

- **ecvl**

## Macros

- #define **ECVL\_TUPLE**(name, ...) name,
- #define **ECVL\_TUPLE**(name, size, type, ...) template<> struct TypeInfo<ecvl::DataType::name> { using basetype = type; };
- #define **ECVL\_TUPLE**(name, ...) + 1
- #define **ECVL\_TUPLE**(name, ...) + 1

## Enumerations

- enum **ecvl::DataType** {  
**ecvl::DataType::ECVL\_TUPLE**, **ecvl::DataType::int8**, **ecvl::DataType::int16**, **ecvl::DataType::int32**,  
**ecvl::DataType::int64**, **ecvl::DataType::float32**, **ecvl::DataType::float64**, **ecvl::DataType::uint8**,  
**ecvl::DataType::uint16**, **ecvl::DataType::uint32**, **ecvl::DataType::uint64**, **ecvl::DataType::none** }

*DataType is an enum class which defines data types allowed for images.*

## Functions

- uint8\_t **ecvl::DataTypeSize** (DataType dt)  
*Provides the size in bytes of a given DataType.*
- constexpr size\_t **ecvl::DataTypeSize** ()  
*Function to get the number of existing DataType at compile time.*
- constexpr size\_t **ecvl::DataTypeSignedSize** ()  
*Function to get the number of existing signed DataType at compile time.*
- constexpr std::array< DataType, DataTypeSize()> **ecvl::DataTypeArray** ()  
*Function to get a std::array with all the DataType values at compile time.*
- constexpr std::array< DataType, DataTypeSignedSize()> **ecvl::DataTypeSignedArray** ()  
*Function to get a std::array with all the signed DataType values at compile time.*

### 9.6.1 Macro Definition Documentation

#### 9.6.1.1 ECVL\_TUPLE [1/4]

```
#define ECVL_TUPLE(  
    name,  
    ... ) name,
```

Definition at line 34 of file datatype.h.

#### 9.6.1.2 ECVL\_TUPLE [2/4]

```
#define ECVL_TUPLE(  
    name,  
    size,  
    type,  
    ... ) template<> struct TypeInfo<ecvl::DataType::name> { using basetype =  
type; };
```

Definition at line 34 of file datatype.h.



## 9.6.1.3 ECVL\_TUPLE [3/4]

```
#define ECVL_TUPLE(
    name,
    ... ) + 1
```

Definition at line 34 of file datatype.h.

## 9.6.1.4 ECVL\_TUPLE [4/4]

```
#define ECVL_TUPLE(
    name,
    ... ) + 1
```

Definition at line 34 of file datatype.h.

## 9.7 datatype\_existing\_tuples.inc.h File Reference

```
#include "datatype_existing_tuples_signed.inc.h"
#include "datatype_existing_tuples_unsigned.inc.h"
```

## 9.8 datatype\_existing\_tuples\_signed.inc.h File Reference

## 9.9 datatype\_existing\_tuples\_unsigned.inc.h File Reference

## 9.10 datatype\_matrix.h File Reference

```
#include "datatype.h"
```

## Classes

- struct **ecvl::Table1D**< \_StructFun, Args >
- struct **ecvl::Table1D**< \_StructFun, Args >::integer< i >
- struct **ecvl::SignedTable1D**< \_StructFun, Args >
- struct **ecvl::SignedTable1D**< \_StructFun, Args >::integer< i >
- struct **ecvl::Table2D**< \_StructFun, Args >
- struct **ecvl::Table2D**< \_StructFun, Args >::integer< i >

## Namespaces

- **ecvl**

## 9.11 datatype\_tuples.inc.h File Reference

```
#include "datatype_existing_tuples.inc.h"
```

## 9.12 filesystem.cc File Reference

```
#include "ecvl/core/filesystem.h"  
#include <algorithm>  
#include <fstream>  
#include <string>  
#include <sys/stat.h>  
#include <sys/types.h>
```

### Namespaces

- **filesystem**

### Functions

- bool **filesystem::exists** (const path &p)
- bool **filesystem::exists** (const path &p, error\_code &ec)
- bool **filesystem::create\_directories** (const path &p)
- bool **filesystem::create\_directories** (const path &p, error\_code &ec)
- void **filesystem::copy** (const path &from, const path &to)
- void **filesystem::copy** (const path &from, const path &to, error\_code &ec)

## 9.13 filesystem.h File Reference

```
#include <string>  
#include <system_error>
```

### Classes

- class **filesystem::path**

### Namespaces

- **filesystem**

## Functions

- path **filesystem::operator/** (const path &lhs, const path &rhs)
- bool **filesystem::exists** (const path &p)
- bool **filesystem::exists** (const path &p, std::error\_code &ec)
- bool **filesystem::create\_directories** (const path &p)
- bool **filesystem::create\_directories** (const path &p, std::error\_code &ec)
- void **filesystem::copy** (const path &from, const path &to)
- void **filesystem::copy** (const path &from, const path &to, std::error\_code &ec)

## 9.14 gui.cpp File Reference

```
#include "ecvl/gui.h"
```

## Namespaces

- **ecvl**

## Functions

- wxImage **ecvl::wx\_from\_mat** (Image &img)
- void **ecvl::ImShow** (const Image &img)

*Displays an **Image** (p. 60).*

## 9.15 gui.h File Reference

```
#include <wx/wx.h>
#include "ecvl/core/image.h"
```

## Classes

- class **ecvl::wxImagePanel**  
***wxImagePanel** (p. 107) creates a wxPanel to contain an **Image** (p. 60).*
- class **ecvl::ShowApp**  
***ShowApp** (p. 80) is a custom wxApp which allows you to visualize an ECVL **Image** (p. 60).*

## Namespaces

- **ecvl**

## Functions

- void **ecvl::ImShow** (const Image &img)  
*Displays an **Image** (p. 60).*

## 9.16 home.h File Reference

## 9.17 image.cpp File Reference

```
#include "ecvl/core/image.h"
#include "ecvl/core/datatype_matrix.h"
#include "ecvl/core/standard_errors.h"
```

### Namespaces

- **ecvl**

### Functions

- void **ecvl::RearrangeChannels** (const Image &src, Image &dst, const std::string &channels)  
*Changes the order of the **Image** (p. 60) dimensions.*
- void **ecvl::CopyImage** (const Image &src, Image &dst, DataType new\_type=DataType::none)  
*Copies the source **Image** (p. 60) into the destination **Image** (p. 60).*

## 9.18 image.h File Reference

```
#include <algorithm>
#include <numeric>
#include <stdexcept>
#include <vector>
#include <opencv2/core.hpp>
#include "datatype.h"
#include "memorymanager.h"
#include "iterators.h"
#include "iterators_impl.inc.h"
```

### Classes

- class **ecvl::MetaData**
- class **ecvl::Image**  
***Image** (p. 60) class.*
- class **ecvl::View**< DT >
- class **ecvl::ConstView**< DT >
- class **ecvl::ContiguousView**< DT >
- class **ecvl::ConstContiguousView**< DT >
- class **ecvl::ContiguousViewXYC**< DT >
- struct **ecvl::StructCopyImage**< SDT, DDT >

### Namespaces

- **ecvl**

## Enumerations

- enum **ecvl::ColorType** {  
**ecvl::ColorType::none**, **ecvl::ColorType::GRAY**, **ecvl::ColorType::RGB**, **ecvl::ColorType::BGR**,  
**ecvl::ColorType::HSV**, **ecvl::ColorType::YCbCr** }

*Enum class representing the ECVL supported color spaces.*

## Functions

- void **ecvl::RearrangeChannels** (const Image &src, Image &dst, const std::string &channels)  
*Changes the order of the **Image** (p. 60) dimensions.*
- void **ecvl::CopyImage** (const Image &src, Image &dst, DataType new\_type=DataType::none)  
*Copies the source **Image** (p. 60) into the destination **Image** (p. 60).*

## 9.19 imgcodecs.cpp File Reference

```
#include "ecvl/core/imgcodecs.h"
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include "ecvl/core/support_opencv.h"
```

## Namespaces

- ecvl**

## Functions

- bool **ecvl::ImRead** (const std::string &filename, Image &dst)  
*Loads an image from a file.*
- bool **ecvl::ImRead** (const **filesystem::path** &filename, Image &dst)
- bool **ecvl::ImWrite** (const std::string &filename, const Image &src)  
*Saves an image into a specified file.*
- bool **ecvl::ImWrite** (const **filesystem::path** &filename, const Image &src)

## 9.20 imgcodecs.h File Reference

```
#include <string>
#include "image.h"
#include "filesystem.h"
```

## Namespaces

- ecvl**

## Functions

- bool **ecvl::ImRead** (const std::string &filename, Image &dst)  
*Loads an image from a file.*
- bool **ecvl::ImRead** (const **filesystem::path** &filename, Image &dst)
- bool **ecvl::ImWrite** (const std::string &filename, const Image &src)  
*Saves an image into a specified file.*
- bool **ecvl::ImWrite** (const **filesystem::path** &filename, const Image &src)

## 9.21 imgproc.cpp File Reference

```
#include "ecvl/core/imgproc.h"
#include <stdexcept>
#include <opencv2/imgproc.hpp>
#include "ecvl/core/datatype_matrix.h"
#include "ecvl/core/standard_errors.h"
```

## Namespaces

- **ecvl**

## Functions

- void **ecvl::ResizeDim** (const **ecvl::Image** &src, **ecvl::Image** &dst, const std::vector< int > &newdims, InterpolationType interp=InterpolationType::linear)  
*Resizes an **Image** (p. 60) to a new dimension.*
- void **ecvl::ResizeScale** (const **ecvl::Image** &src, **ecvl::Image** &dst, const std::vector< double > &scales, InterpolationType interp=InterpolationType::linear)  
*Resizes an **Image** (p. 60) by scaling the dimentions to a given scale factor.*
- void **ecvl::Flip2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Flips an **Image** (p. 60).*
- void **ecvl::Mirror2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Mirrors an **Image** (p. 60).*
- void **ecvl::Rotate2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, const std::vector< double > &center={}, double scale=1.0, InterpolationType interp=InterpolationType::linear)  
*Rotates an **Image** (p. 60).*
- void **ecvl::RotateFullImage2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, double scale=1.0, InterpolationType interp=InterpolationType::linear)  
*Rotates an **Image** (p. 60) resizing the output accordingly.*
- void **ecvl::ChangeColorSpace** (const Image &src, Image &dst, ColorType new\_type)  
*Copies the source **Image** (p. 60) into destination **Image** (p. 60) changing the color space.*
- void **ecvl::Threshold** (const Image &src, Image &dst, double thresh, double maxval, ThresholdingType thresh\_type=ThresholdingType::BINARY)  
*Applies a fixed threshold to an input **Image** (p. 60).*
- double **ecvl::OtsuThreshold** (const Image &src)  
*Calculates the Otsu thresholding value.*

## 9.22 imgproc.h File Reference

```
#include "image.h"
#include "support_opencv.h"
```

### Namespaces

- **ecvl**

### Enumerations

- enum **ecvl::ThresholdingType** { **ecvl::ThresholdingType::BINARY**, **ecvl::ThresholdingType::BINARY\_<math>\leftrightarrow</math>Y\_INV** }  
*Enum class representing the ECVL thresholding types.*
- enum **ecvl::InterpolationType** { **ecvl::InterpolationType::nearest**, **ecvl::InterpolationType::linear**, **ecvl::InterpolationType::area**, **ecvl::InterpolationType::cubic**, **ecvl::InterpolationType::lanczos4** }  
*Enum class representing the ECVL interpolation types.*

### Functions

- void **ecvl::ResizeDim** (const **ecvl::Image** &src, **ecvl::Image** &dst, const std::vector< int > &newdims, InterpolationType interp=InterpolationType::linear)  
*Resizes an **Image** (p. 60) to a new dimension.*
- void **ecvl::ResizeScale** (const **ecvl::Image** &src, **ecvl::Image** &dst, const std::vector< double > &scales, InterpolationType interp=InterpolationType::linear)  
*Resizes an **Image** (p. 60) by scaling the dimensions to a given scale factor.*
- void **ecvl::Flip2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Flips an **Image** (p. 60).*
- void **ecvl::Mirror2D** (const **ecvl::Image** &src, **ecvl::Image** &dst)  
*Mirrors an **Image** (p. 60).*
- void **ecvl::Rotate2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, const std::vector< double > &center={}, double scale=1.0, InterpolationType interp=InterpolationType::linear)  
*Rotates an **Image** (p. 60).*
- void **ecvl::RotateFullImage2D** (const **ecvl::Image** &src, **ecvl::Image** &dst, double angle, double scale=1.0, InterpolationType interp=InterpolationType::linear)  
*Rotates an **Image** (p. 60) resizing the output accordingly.*
- void **ecvl::ChangeColorSpace** (const **Image** &src, **Image** &dst, ColorType new\_type)  
*Copies the source **Image** (p. 60) into destination **Image** (p. 60) changing the color space.*
- void **ecvl::Threshold** (const **Image** &src, **Image** &dst, double thresh, double maxval, ThresholdingType thresh\_type=ThresholdingType::BINARY)  
*Applies a fixed threshold to an input **Image** (p. 60).*
- double **ecvl::OtsuThreshold** (const **Image** &src)  
*Calculates the Otsu thresholding value.*

## 9.23 iterators.h File Reference

```
#include <vector>
#include <cstdint>
```

### Classes

- struct **ecvl::Iterator**< T >
- struct **ecvl::ConstIterator**< T >
- struct **ecvl::ContiguousIterator**< T >
- struct **ecvl::ConstContiguousIterator**< T >

### Namespaces

- **ecvl**

## 9.24 iterators\_impl.inc.h File Reference

## 9.25 memorymanager.cpp File Reference

```
#include "ecvl/core/memorymanager.h"
```

## 9.26 memorymanager.h File Reference

```
#include <cstdint>
#include <cstring>
#include <stdexcept>
```

### Classes

- class **MemoryManager**
- class **DefaultMemoryManager**
- class **ShallowMemoryManager**



## 9.27 standard\_errors.h File Reference

### Macros

- `#define ECVL_ERROR_MSG "[Error]: "`
- `#define ECVL_WARNING_MSG "[Warning]: "`
- `#define ECVL_ERROR_NOT_IMPLEMENTED throw std::runtime_error( ECVL_ERROR_MSG "Not implemented");`
- `#define ECVL_ERROR_NOT_REACHABLE_CODE throw std::runtime_error( ECVL_ERROR_MSG "How did you get here?");`
- `#define ECVL_ERROR_WRONG_PARAMS(...) throw std::runtime_error( ECVL_ERROR_MSG "Wrong parameters - " __VA_ARGS__);`
- `#define ECVL_ERROR_NOT_ALLOWED_ON_NON_OWING_IMAGE(...) throw std::runtime_error( ECVL_ERROR_MSG "Operation not allowed on non-owning Image" __VA_ARGS__);`
- `#define ECVL_ERROR_UNSUPPORTED_OPENCV_DEPTH throw std::runtime_error( ECVL_ERROR_MSG "Unsupported OpenCV depth");`
- `#define ECVL_ERROR_UNSUPPORTED_OPENCV_DIMS throw std::runtime_error( ECVL_ERROR_MSG "Unsupported OpenCV dimensions");`
- `#define ECVL_ERROR_EMPTY_IMAGE throw std::runtime_error( ECVL_ERROR_MSG "Empty image provided");`
- `#define ECVL_ERROR_NOT_ALLOWED_ON_UNSIGNED_IMG throw std::runtime_error( ECVL_ERROR_MSG "Operation not allowed on unsigned Image");`

### 9.27.1 Macro Definition Documentation

#### 9.27.1.1 ECVL\_ERROR\_EMPTY\_IMAGE

```
#define ECVL_ERROR_EMPTY_IMAGE throw std::runtime_error( ECVL_ERROR_MSG "Empty image provided");
```

Definition at line 13 of file `standard_errors.h`.

#### 9.27.1.2 ECVL\_ERROR\_MSG

```
#define ECVL_ERROR_MSG "[Error]: "
```

Definition at line 4 of file `standard_errors.h`.

#### 9.27.1.3 ECVL\_ERROR\_NOT\_ALLOWED\_ON\_NON\_OWING\_IMAGE

```
#define ECVL_ERROR_NOT_ALLOWED_ON_NON_OWING_IMAGE(  
    ... ) throw std::runtime_error( ECVL_ERROR_MSG "Operation not allowed on non-owning  
Image" __VA_ARGS__);
```

Definition at line 10 of file `standard_errors.h`.

#### 9.27.1.4 ECVL\_ERROR\_NOT\_ALLOWED\_ON\_UNSIGNED\_IMG

```
#define ECVL_ERROR_NOT_ALLOWED_ON_UNSIGNED_IMG throw std::runtime_error( ECVL_ERROR_MSG "Operation  
not allowed on unsigned Image");
```

Definition at line 14 of file standard\_errors.h.

#### 9.27.1.5 ECVL\_ERROR\_NOT\_IMPLEMENTED

```
#define ECVL_ERROR_NOT_IMPLEMENTED throw std::runtime_error( ECVL_ERROR_MSG "Not implemented");
```

Definition at line 7 of file standard\_errors.h.

#### 9.27.1.6 ECVL\_ERROR\_NOT\_REACHABLE\_CODE

```
#define ECVL_ERROR_NOT_REACHABLE_CODE throw std::runtime_error( ECVL_ERROR_MSG "How did you  
get here?");
```

Definition at line 8 of file standard\_errors.h.

#### 9.27.1.7 ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DEPTH

```
#define ECVL_ERROR_UNSUPPORTED_OPENCV_DEPTH throw std::runtime_error( ECVL_ERROR_MSG "Unsupported  
OpenCV depth");
```

Definition at line 11 of file standard\_errors.h.

#### 9.27.1.8 ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DIMS

```
#define ECVL_ERROR_UNSUPPORTED_OPENCV_DIMS throw std::runtime_error( ECVL_ERROR_MSG "Unsupported  
OpenCV dimensions");
```

Definition at line 12 of file standard\_errors.h.

#### 9.27.1.9 ECVL\_ERROR\_WRONG\_PARAMS

```
#define ECVL_ERROR_WRONG_PARAMS(  
    ... ) throw std::runtime_error( ECVL_ERROR_MSG "Wrong parameters - " __VA_ARGS↵  
    );
```

Definition at line 9 of file standard\_errors.h.

## 9.27.1.10 ECVL\_WARNING\_MSG

```
#define ECVL_WARNING_MSG "[Warning]: "
```

Definition at line 5 of file standard\_errors.h.

## 9.28 support\_opencv.cpp File Reference

```
#include "ecvl/core/support_opencv.h"  
#include "ecvl/core/standard_errors.h"
```

### Namespaces

- **ecvl**

### Functions

- **ecvl::Image ecvl::MatToImage** (const cv::Mat &m)  
*Convert a cv::Mat into an **ecvl::Image** (p. 60).*
- cv::Mat **ecvl::ImageToMat** (const Image &img)  
*Convert an ECVL **Image** (p. 60) into OpenCV Mat.*

## 9.29 support\_opencv.h File Reference

```
#include "image.h"
```

### Namespaces

- **ecvl**

### Functions

- **ecvl::Image ecvl::MatToImage** (const cv::Mat &m)  
*Convert a cv::Mat into an **ecvl::Image** (p. 60).*
- cv::Mat **ecvl::ImageToMat** (const Image &img)  
*Convert an ECVL **Image** (p. 60) into OpenCV Mat.*

## 9.30 test\_core.cpp File Reference

```
#include <gtest/gtest.h>  
#include "ecvl/core.h"
```

## Functions

- **TEST** (Core, CreateEmptyImage)
- **TEST** (Core, CreateImageWithFiveDims)

### 9.30.1 Function Documentation

#### 9.30.1.1 TEST() [1/2]

```
TEST (
    Core ,
    CreateEmptyImage )
```

Definition at line 7 of file test\_core.cpp.

#### 9.30.1.2 TEST() [2/2]

```
TEST (
    Core ,
    CreateImageWithFiveDims )
```

Definition at line 14 of file test\_core.cpp.

# Index

- - ecvl::DivImpl< Image, Image, ET >, 58
  - ecvl::DivImpl< Image, T, ET >, 59
  - ecvl::DivImpl< ST1, ST2, ET >, 57
  - ecvl::DivImpl< T, Image, ET >, 59
  - ecvl::StructAdd< a, b >, 83
  - ecvl::StructCopyImage< SDT, DDT >, 84
  - ecvl::StructDiv< a, b, ET >, 85
  - ecvl::StructMul< a, b >, 86
  - ecvl::StructScalarAdd< DT, T >, 86
  - ecvl::StructScalarDiv< DT, T >, 87
  - ecvl::StructScalarDivInv< DT, T, ET >, 88
  - ecvl::StructScalarMul< DT, T >, 89
  - ecvl::StructScalarNeg< DT >, 89
  - ecvl::StructScalarSub< DT, T >, 90
  - ecvl::StructScalarSubInv< DT, T >, 91
  - ecvl::StructSub< a, b >, 92
- ~Image
  - ecvl::Image, 63
- ~MemoryManager
  - MemoryManager, 74
- ~MetaData
  - ecvl::MetaData, 75
- Add
  - ecvl, 18, 19
- Allocate
  - DefaultMemoryManager, 56
  - MemoryManager, 74
  - ShallowMemoryManager, 79
- AllocateAndCopy
  - DefaultMemoryManager, 56
  - MemoryManager, 74
  - ShallowMemoryManager, 79
- area
  - ecvl, 17
- arithmetic.cpp, 109
  - STANDARD\_INPLACE\_OPERATION, 110
  - STANDARD\_NON\_INPLACE\_OPERATION, 110
- arithmetic.h, 110
- basetype
  - ecvl::ConstContiguousView< DT >, 42
  - ecvl::ConstView< DT >, 47
  - ecvl::ContiguousView< DT >, 52
  - ecvl::ContiguousViewXYC< DT >, 54
  - ecvl::TypeInfo< DataType >, 97
  - ecvl::TypeInfo< ecvl::DataType::float32 >, 97
  - ecvl::TypeInfo< ecvl::DataType::float64 >, 98
  - ecvl::TypeInfo< ecvl::DataType::int16 >, 99
  - ecvl::TypeInfo< ecvl::DataType::int32 >, 99
  - ecvl::TypeInfo< ecvl::DataType::int64 >, 100
  - ecvl::TypeInfo< ecvl::DataType::int8 >, 101
  - ecvl::TypeInfo< ecvl::DataType::none >, 101
  - ecvl::TypeInfo< ecvl::DataType::uint16 >, 102
  - ecvl::TypeInfo< ecvl::DataType::uint32 >, 103
  - ecvl::TypeInfo< ecvl::DataType::uint64 >, 103
  - ecvl::TypeInfo< ecvl::DataType::uint8 >, 104
  - ecvl::View< DT >, 105
- Begin
  - ecvl::ConstContiguousView< DT >, 43
  - ecvl::ConstView< DT >, 48
  - ecvl::ContiguousView< DT >, 52
  - ecvl::ContiguousViewXYC< DT >, 54
  - ecvl::Image, 63
  - ecvl::View< DT >, 106
- BGR
  - ecvl, 16
- BINARY
  - ecvl, 18
- BINARY\_INV
  - ecvl, 18
- ChangeColorSpace
  - ecvl, 19
- channels
  - ecvl::ContiguousViewXYC< DT >, 54
- channels\_
  - ecvl::Image, 67
- ColorType
  - ecvl, 16
- colortype\_
  - ecvl::Image, 67
- ConstContiguousIterator
  - ecvl::ConstContiguousIterator< T >, 39
- ConstContiguousView
  - ecvl::ConstContiguousView< DT >, 42
- ConstIterator
  - ecvl::ConstIterator< T >, 44
- ConstView
  - ecvl::ConstView< DT >, 47
- contiguous\_
  - ecvl::Image, 67
- ContiguousBegin
  - ecvl::Image, 63, 64
- ContiguousEnd
  - ecvl::Image, 64
- ContiguousIterator
  - ecvl::ContiguousIterator< T >, 49
- ContiguousView

- ecvl::ContiguousView< DT >, 52
- ContiguousViewXYC
  - ecvl::ContiguousViewXYC< DT >, 54
- copy
  - filesystem, 35, 36
- CopyImage
  - ecvl, 20
- core.cpp, 112
- core.h, 112
- Create
  - ecvl::Image, 64
- create\_directories
  - filesystem, 36
- cubic
  - ecvl, 17
- data
  - ecvl::SignedTable1D< \_StructFun, Args >, 83
  - ecvl::Table1D< \_StructFun, Args >, 94
  - ecvl::Table2D< \_StructFun, Args >, 96
- data\_
  - ecvl::Image, 68
- datasize\_
  - ecvl::Image, 68
- DataType
  - ecvl, 16
- datatype.cpp, 112
  - ECVL\_TUPLE, 113
- datatype.h, 113
  - ECVL\_TUPLE, 114, 115
- datatype\_existing\_tuples.inc.h, 115
- datatype\_existing\_tuples\_signed.inc.h, 115
- datatype\_existing\_tuples\_unsigned.inc.h, 115
- datatype\_matrix.h, 115
- datatype\_tuples.inc.h, 116
- DataTypeArray
  - ecvl, 21
- DataTypeSignedArray
  - ecvl, 21
- DataTypeSignedSize
  - ecvl, 21
- DataTypeSize
  - ecvl, 21, 22
- Deallocate
  - DefaultMemoryManager, 56
  - MemoryManager, 75
  - ShallowMemoryManager, 79
- DefaultMemoryManager, 55
  - Allocate, 56
  - AllocateAndCopy, 56
  - Deallocate, 56
  - GetInstance, 56
- dims\_
  - ecvl::Image, 68
- Div
  - ecvl, 22, 23
- ecvl, 13
  - Add, 18, 19
  - area, 17
  - BGR, 16
  - BINARY, 18
  - BINARY\_INV, 18
  - ChangeColorSpace, 19
  - ColorType, 16
  - CopyImage, 20
  - cubic, 17
  - DataType, 16
  - DataTypeArray, 21
  - DataTypeSignedArray, 21
  - DataTypeSignedSize, 21
  - DataTypeSize, 21, 22
  - Div, 22, 23
  - ECVL\_TUPLE, 17
  - Flip2D, 23
  - float32, 17
  - float64, 17
  - GRAY, 16
  - HSV, 16
  - ImageToMat, 23
  - ImRead, 23, 24
  - ImShow, 24
  - ImWrite, 25
  - int16, 17
  - int32, 17
  - int64, 17
  - int8, 17
  - InterpolationType, 17
  - lanczos4, 17
  - linear, 17
  - MatToImage, 26
  - Mirror2D, 26
  - Mul, 26, 27
  - nearest, 17
  - Neg, 28
  - none, 16, 17
  - OtsuThreshold, 29
  - RearrangeChannels, 29
  - ResizeDim, 29
  - ResizeScale, 30
  - RGB, 16
  - Rotate2D, 30
  - RotateFullImage2D, 31
  - saturate\_cast, 31, 32
  - Sub, 32–34
  - Threshold, 34
  - ThresholdingType, 17
  - uint16, 17
  - uint32, 17
  - uint64, 17
  - uint8, 17
  - wx\_from\_mat, 35
  - YCbCr, 16
- ecvl::ConstContiguousIterator< T >, 39
  - ConstContiguousIterator, 39
  - img\_, 41
  - operator \*, 40

- operator!=, 40
- operator++, 40
- operator->, 40
- operator==, 40
- ptr\_, 41
- ecvl::ConstContiguousView< DT >, 41
  - basetype, 42
  - Begin, 43
  - ConstContiguousView, 42
  - End, 43
  - operator(), 43
- ecvl::ConstIterator< T >, 43
  - ConstIterator, 44
  - img\_, 46
  - IncrementMemFn, 44
  - incrementor, 46
  - operator \*, 45
  - operator!=, 45
  - operator++, 45
  - operator->, 45
  - operator==, 45
  - pos\_, 46
  - ptr\_, 46
- ecvl::ConstView< DT >, 47
  - basetype, 47
  - Begin, 48
  - ConstView, 47
  - End, 48
  - operator(), 48
- ecvl::ContiguousIterator< T >, 48
  - ContiguousIterator, 49
  - img\_, 50
  - operator \*, 49
  - operator!=, 49
  - operator++, 50
  - operator->, 50
  - operator==, 50
  - ptr\_, 50
- ecvl::ContiguousView< DT >, 51
  - basetype, 52
  - Begin, 52
  - ContiguousView, 52
  - End, 52
  - operator(), 52
- ecvl::ContiguousViewXYC< DT >, 53
  - basetype, 54
  - Begin, 54
  - channels, 54
  - ContiguousViewXYC, 54
  - End, 54
  - height, 55
  - operator(), 55
  - width, 55
- ecvl::DivImpl< Image, Image, ET >, 58
  - \_, 58
- ecvl::DivImpl< Image, T, ET >, 58
  - \_, 59
- ecvl::DivImpl< ST1, ST2, ET >, 57
  - \_, 57
- ecvl::DivImpl< T, Image, ET >, 59
  - \_, 59
- ecvl::Image, 60
  - ~Image, 63
  - Begin, 63
  - channels\_, 67
  - colortype\_, 67
  - contiguous\_, 67
  - ContiguousBegin, 63, 64
  - ContiguousEnd, 64
  - Create, 64
  - data\_, 68
  - datasize\_, 68
  - dims\_, 68
  - elemsize\_, 68
  - elemtype\_, 68
  - End, 65
  - Image, 62
  - IsEmpty, 65
  - IsOwner, 66
  - mem\_, 69
  - meta\_, 69
  - operator=, 66
  - Ptr, 66
  - strides\_, 69
  - swap, 67
- ecvl::Iterator< T >, 71
  - img\_, 73
  - IncrementMemFn, 71
  - incrementor, 73
  - Iterator, 71
  - operator \*, 72
  - operator!=, 72
  - operator++, 72
  - operator->, 72
  - operator==, 72
  - pos\_, 73
  - ptr\_, 73
- ecvl::MetaData, 75
  - ~MetaData, 75
  - Query, 76
- ecvl::ShowApp, 80
  - OnInit, 80
  - ShowApp, 80
- ecvl::SignedTable1D< \_StructFun, Args >, 81
  - data, 83
  - FillData, 82
  - fun\_type, 82
  - operator(), 82
  - SignedTable1D, 82
- ecvl::SignedTable1D< \_StructFun, Args >::integer< i >, 70
- ecvl::StructAdd< a, b >, 83
  - \_, 83
- ecvl::StructCopyImage< SDT, DDT >, 84
  - \_, 84
- ecvl::StructDiv< a, b, ET >, 85

- \_, 85
- ecvl::StructMul< a, b >, 85
- \_, 86
- ecvl::StructScalarAdd< DT, T >, 86
- \_, 86
- ecvl::StructScalarDiv< DT, T >, 87
- \_, 87
- ecvl::StructScalarDivInv< DT, T, ET >, 87
- \_, 88
- ecvl::StructScalarMul< DT, T >, 88
- \_, 89
- ecvl::StructScalarNeg< DT >, 89
- \_, 89
- ecvl::StructScalarSub< DT, T >, 90
- \_, 90
- ecvl::StructScalarSubInv< DT, T >, 91
- \_, 91
- ecvl::StructSub< a, b >, 91
- \_, 92
- ecvl::Table1D< \_StructFun, Args >, 92
  - data, 94
  - FillData, 93
  - fun\_type, 93
  - operator(), 93
  - Table1D, 93
- ecvl::Table1D< \_StructFun, Args >::integer< i >, 70
- ecvl::Table2D< \_StructFun, Args >, 94
  - data, 96
  - FillData, 95
  - fun\_type, 95
  - operator(), 96
  - Table2D, 95
- ecvl::Table2D< \_StructFun, Args >::integer< i >, 70
- ecvl::TypeInfo< DataType >, 96
  - basetype, 97
- ecvl::TypeInfo< ecvl::DataType::float32 >, 97
  - basetype, 97
- ecvl::TypeInfo< ecvl::DataType::float64 >, 98
  - basetype, 98
- ecvl::TypeInfo< ecvl::DataType::int16 >, 98
  - basetype, 99
- ecvl::TypeInfo< ecvl::DataType::int32 >, 99
  - basetype, 99
- ecvl::TypeInfo< ecvl::DataType::int64 >, 100
  - basetype, 100
- ecvl::TypeInfo< ecvl::DataType::int8 >, 100
  - basetype, 101
- ecvl::TypeInfo< ecvl::DataType::none >, 101
  - basetype, 101
- ecvl::TypeInfo< ecvl::DataType::uint16 >, 102
  - basetype, 102
- ecvl::TypeInfo< ecvl::DataType::uint32 >, 102
  - basetype, 103
- ecvl::TypeInfo< ecvl::DataType::uint64 >, 103
  - basetype, 103
- ecvl::TypeInfo< ecvl::DataType::uint8 >, 104
  - basetype, 104
- ecvl::View< DT >, 104
- basetype, 105
- Begin, 106
- End, 106
- operator(), 106
- View, 105
- ecvl::wxImagePanel, 107
  - SetImage, 107
  - wxImagePanel, 107
- ECVL\_ERROR\_EMPTY\_IMAGE
  - standard\_errors.h, 123
- ECVL\_ERROR\_MSG
  - standard\_errors.h, 123
- ECVL\_ERROR\_NOT\_ALLOWED\_ON\_NON\_OWING\_IMAGE
  - standard\_errors.h, 123
- ECVL\_ERROR\_NOT\_ALLOWED\_ON\_UNSIGNED\_IMG
  - standard\_errors.h, 123
- ECVL\_ERROR\_NOT\_IMPLEMENTED
  - standard\_errors.h, 124
- ECVL\_ERROR\_NOT\_REACHABLE\_CODE
  - standard\_errors.h, 124
- ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DEPTH
  - standard\_errors.h, 124
- ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DIMS
  - standard\_errors.h, 124
- ECVL\_ERROR\_WRONG\_PARAMS
  - standard\_errors.h, 124
- ECVL\_TUPLE
  - datatype.cpp, 113
  - datatype.h, 114, 115
  - ecvl, 17
- ECVL\_WARNING\_MSG
  - standard\_errors.h, 124
- elemsize\_
  - ecvl::Image, 68
- elemtype\_
  - ecvl::Image, 68
- End
  - ecvl::ConstContiguousView< DT >, 43
  - ecvl::ConstView< DT >, 48
  - ecvl::ContiguousView< DT >, 52
  - ecvl::ContiguousViewXYC< DT >, 54
  - ecvl::Image, 65
  - ecvl::View< DT >, 106
- exists
  - filesystem, 37
- filesystem, 35
  - copy, 35, 36
  - create\_directories, 36
  - exists, 37
  - operator/, 37
- filesystem.cc, 116
- filesystem.h, 116
- filesystem::path, 76
  - operator=, 77
  - operator=, 77
  - parent\_path, 77
  - path, 76
  - stem, 77



- string, 78
- FillData
  - ecvl::SignedTable1D< \_StructFun, Args >, 82
  - ecvl::Table1D< \_StructFun, Args >, 93
  - ecvl::Table2D< \_StructFun, Args >, 95
- Flip2D
  - ecvl, 23
- float32
  - ecvl, 17
- float64
  - ecvl, 17
- fun\_type
  - ecvl::SignedTable1D< \_StructFun, Args >, 82
  - ecvl::Table1D< \_StructFun, Args >, 93
  - ecvl::Table2D< \_StructFun, Args >, 95
- GetInstance
  - DefaultMemoryManager, 56
  - ShallowMemoryManager, 79
- GRAY
  - ecvl, 16
- gui.cpp, 117
- gui.h, 117
- height
  - ecvl::ContiguousViewXYC< DT >, 55
- home.h, 118
- HSV
  - ecvl, 16
- Image
  - ecvl::Image, 62
- image.cpp, 118
- image.h, 118
- ImageToMat
  - ecvl, 23
- img\_
  - ecvl::ConstContiguousIterator< T >, 41
  - ecvl::ConstIterator< T >, 46
  - ecvl::ContiguousIterator< T >, 50
  - ecvl::Iterator< T >, 73
- imgcodecs.cpp, 119
- imgcodecs.h, 119
- imgproc.cpp, 120
- imgproc.h, 121
- ImRead
  - ecvl, 23, 24
- ImShow
  - ecvl, 24
- ImWrite
  - ecvl, 25
- IncrementMemFn
  - ecvl::ConstIterator< T >, 44
  - ecvl::Iterator< T >, 71
- incrementor
  - ecvl::ConstIterator< T >, 46
  - ecvl::Iterator< T >, 73
- int16
  - ecvl, 17
- int32
  - ecvl, 17
- int64
  - ecvl, 17
- int8
  - ecvl, 17
- InterpolationType
  - ecvl, 17
- IsEmpty
  - ecvl::Image, 65
- IsOwner
  - ecvl::Image, 66
- Iterator
  - ecvl::Iterator< T >, 71
- iterators.h, 122
- iterators\_impl.inc.h, 122
- lanczos4
  - ecvl, 17
- linear
  - ecvl, 17
- MatToImage
  - ecvl, 26
- mem\_
  - ecvl::Image, 69
- MemoryManager, 74
  - ~MemoryManager, 74
  - Allocate, 74
  - AllocateAndCopy, 74
  - Deallocate, 75
- memorymanager.cpp, 122
- memorymanager.h, 122
- meta\_
  - ecvl::Image, 69
- Mirror2D
  - ecvl, 26
- Mul
  - ecvl, 26, 27
- nearest
  - ecvl, 17
- Neg
  - ecvl, 28
- none
  - ecvl, 16, 17
- OnInit
  - ecvl::ShowApp, 80
- operator \*
  - ecvl::ConstContiguousIterator< T >, 40
  - ecvl::ConstIterator< T >, 45
  - ecvl::ContiguousIterator< T >, 49
  - ecvl::Iterator< T >, 72
- operator!=
  - ecvl::ConstContiguousIterator< T >, 40
  - ecvl::ConstIterator< T >, 45
  - ecvl::ContiguousIterator< T >, 49
  - ecvl::Iterator< T >, 72

- operator()
  - ecvl::ConstContiguousView< DT >, 43
  - ecvl::ConstView< DT >, 48
  - ecvl::ContiguousView< DT >, 52
  - ecvl::ContiguousViewXYC< DT >, 55
  - ecvl::SignedTable1D< \_StructFun, Args >, 82
  - ecvl::Table1D< \_StructFun, Args >, 93
  - ecvl::Table2D< \_StructFun, Args >, 96
  - ecvl::View< DT >, 106
- operator++
  - ecvl::ConstContiguousIterator< T >, 40
  - ecvl::ConstIterator< T >, 45
  - ecvl::ContiguousIterator< T >, 50
  - ecvl::Iterator< T >, 72
- operator->
  - ecvl::ConstContiguousIterator< T >, 40
  - ecvl::ConstIterator< T >, 45
  - ecvl::ContiguousIterator< T >, 50
  - ecvl::Iterator< T >, 72
- operator/
  - filesystem, 37
- operator/=
  - filesystem::path, 77
- operator=
  - ecvl::Image, 66
  - filesystem::path, 77
- operator==
  - ecvl::ConstContiguousIterator< T >, 40
  - ecvl::ConstIterator< T >, 45
  - ecvl::ContiguousIterator< T >, 50
  - ecvl::Iterator< T >, 72
- OtsuThreshold
  - ecvl, 29
- parent\_path
  - filesystem::path, 77
- path
  - filesystem::path, 76
- pos\_
  - ecvl::ConstIterator< T >, 46
  - ecvl::Iterator< T >, 73
- Ptr
  - ecvl::Image, 66
- ptr\_
  - ecvl::ConstContiguousIterator< T >, 41
  - ecvl::ConstIterator< T >, 46
  - ecvl::ContiguousIterator< T >, 50
  - ecvl::Iterator< T >, 73
- Query
  - ecvl::MetaData, 76
- RearrangeChannels
  - ecvl, 29
- ResizeDim
  - ecvl, 29
- ResizeScale
  - ecvl, 30
- RGB
  - ecvl, 16
- Rotate2D
  - ecvl, 30
- RotateFullImage2D
  - ecvl, 31
- saturate\_cast
  - ecvl, 31, 32
- SetImage
  - ecvl::wxImagePanel, 107
- ShallowMemoryManager, 78
  - Allocate, 79
  - AllocateAndCopy, 79
  - Deallocate, 79
  - GetInstance, 79
- ShowApp
  - ecvl::ShowApp, 80
- SignedTable1D
  - ecvl::SignedTable1D< \_StructFun, Args >, 82
- standard\_errors.h, 123
  - ECVL\_ERROR\_EMPTY\_IMAGE, 123
  - ECVL\_ERROR\_MSG, 123
  - ECVL\_ERROR\_NOT\_ALLOWED\_ON\_NON\_OWING\_IMAGE, 123
  - ECVL\_ERROR\_NOT\_ALLOWED\_ON\_UNSIGNED\_IMG, 123
  - ECVL\_ERROR\_NOT\_IMPLEMENTED, 124
  - ECVL\_ERROR\_NOT\_REACHABLE\_CODE, 124
  - ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DEPTH, 124
  - ECVL\_ERROR\_UNSUPPORTED\_OPENCV\_DIMS, 124
  - ECVL\_ERROR\_WRONG\_PARAMS, 124
  - ECVL\_WARNING\_MSG, 124
- STANDARD\_INPLACE\_OPERATION
  - arithmetic.cpp, 110
- STANDARD\_NON\_INPLACE\_OPERATION
  - arithmetic.cpp, 110
- stem
  - filesystem::path, 77
- strides\_
  - ecvl::Image, 69
- string
  - filesystem::path, 78
- Sub
  - ecvl, 32–34
- support\_opencv.cpp, 125
- support\_opencv.h, 125
- swap
  - ecvl::Image, 67
- Table1D
  - ecvl::Table1D< \_StructFun, Args >, 93
- Table2D
  - ecvl::Table2D< \_StructFun, Args >, 95
- TEST
  - test\_core.cpp, 126
- test\_core.cpp, 125
  - TEST, 126

---

Threshold  
    ecvl, 34

ThresholdingType  
    ecvl, 17

uint16  
    ecvl, 17

uint32  
    ecvl, 17

uint64  
    ecvl, 17

uint8  
    ecvl, 17

View  
    ecvl::View< DT >, 105

width  
    ecvl::ContiguousViewXYC< DT >, 55

wx\_from\_mat  
    ecvl, 35

wxImagePanel  
    ecvl::wxImagePanel, 107

YCbCr  
    ecvl, 16