

■ MODULE : **mamba**

This is the main module of the Mamba Image library. It provides basic functions and classes needed for handling images and defining mathematical morphology transformations and algorithms. This module also contains image display functionalities and other user-friendly features.

■ CLASS :

imageMb

- ◆ `__del__(self)`
- ◆ `__init__(self, *args, **kwargs)`
- ◆ `__str__(self)`
- ◆ `convert(self, depth)`
- ◆ `fastSetPixel(self, value, position)`
- ◆ `fill(self, v)`
- ◆ `freezeDisplay(self)`
- ◆ `getDepth(self)`
- ◆ `getName(self)`
- ◆ `getPixel(self, position)`
- ◆ `getSize(self)`
- ◆ `hideDisplay(self)`
- ◆ `load(self, path, rgbfilter=None)`
- ◆ `reset(self)`
- ◆ `resetPalette(self)`
- ◆ `save(self, path)`
- ◆ `setName(self, name)`
- ◆ `setPalette(self, pal)`
- ◆ `setPixel(self, value, position)`
- ◆ `showDisplay(self)`
- ◆ `unfreezeDisplay(self)`
- ◆ `updateDisplay(self)`

■ FUNCTIONS :

```
def add(imIn1, imIn2, imOut):
def addConst(imIn, v, imOut):
def basinSegment(imIn, imMarker, grid=DEFAULT_GRID, max_level=256):
def buildNeighbor(imMask, imInout, d, grid=DEFAULT_GRID):
def checkEmptiness(imIn):
def compare(imIn1, imIn2, imOut):
def computeDistance(imIn, imOut, grid=DEFAULT_GRID, edge=EMPTY):
def computeMaxRange(imIn):
def computeRange(imIn):
def computeVolume(imIn):
def convert(imIn, imOut):
def convertByMask(imIn, imOut, mFalse, mTrue):
def copy(imIn, imOut):
def copyBitPlane(imIn, plane, imOut):
def copyBytePlane(imIn, plane, imOut):
def copyLine(imIn, nIn, imOut, nOut):
```

```
def cropCopy(imIn, posIn, imOut, posOut, size):
def diff(imIn1, imIn2, imOut):
def diffNeighbor(imIn, imInout, nb, grid=DEFAULT_GRID, edge=EMPTY):
def divConst(imIn, v, imOut):
def dualbuildNeighbor(imMask, imInout, d, grid=DEFAULT_GRID):
def extractFrame(imIn, threshold):
def generateSupMask(imIn1, imIn2, imOut, strict):
def getDirections(grid=DEFAULT_GRID):
def getHistogram(imIn):
def getImageCounter():
def getShowImages():
def gridNeighbors(grid=DEFAULT_GRID):
def hierarBuild(imMask, imInout, grid=DEFAULT_GRID):
def hierarDualBuild(imMask, imInout, grid=DEFAULT_GRID):
def hitOrMiss(imIn, imOut, cse0, cse1, grid=DEFAULT_GRID):
def infFarNeighbor(imIn, imInout, nb, amp, grid=DEFAULT_GRID, edge=FILLED):
def infNeighbor(imIn, imInout, nb, count, grid=DEFAULT_GRID, edge=FILLED):
def label(imIn, imOut, lblow=1, lbhigh=256, grid=DEFAULT_GRID):
def logic(imIn1, imIn2, imOut, log):
def lookup(imIn, imOut, lutable):
def mul(imIn1, imIn2, imOut):
def mulConst(imIn, v, imOut):
def negate(imIn, imOut):
def rotateDirection(d, step=1, grid=DEFAULT_GRID):
def setDefaultGrid(grid):
def setImageIndex(index):
def setShowImages(showThem):
def shift(imIn, imOut, d, amp, fill, grid=DEFAULT_GRID):
def sub(imIn1, imIn2, imOut):
def subConst(imIn, v, imOut):
def supFarNeighbor(imIn, imInout, nb, amp, grid=DEFAULT_GRID, edge=EMPTY):
def supNeighbor(imIn, imInout, nb, count, grid=DEFAULT_GRID, edge=EMPTY):
def threshold(imIn, imOut, low, high):
def tidyDisplays(displayer=None):
def transposeDirection(d, grid=DEFAULT_GRID):
def watershedSegment(imIn, imMarker, grid=DEFAULT_GRID, max_level=256):
```

■ MODULE : **mambaDraw**

This module defines functions to draw inside mamba images. Drawing functions include lines, squares, ... The module also provides functions to extract pixel information.

■ FUNCTIONS :

```
def drawBox(imOut, square, value):
def drawCircle(imOut, circle, value):
def drawFillCircle(imOut, circle, value):
def drawLine(imOut, line, value):
def drawSquare(imOut, square, value):
def getIntensityAlongLine(imOut, line):
```

■ MODULE : **mambaExtra**

This module defines specific functions and classes which can be considered as extras for the Mamba image library. They provide optional display methods, interactive tools and bridges for exchanging images between the Mamba and PIL libraries.

■ FUNCTIONS :

```
def Mamba2PIL(imIn):
```

```
def PIL2Mamba(pilim, imOut):
def dynamicThreshold(imIn):
def hitormissPatternSelector(grid=DEFAULT_GRID):
def mix(imInR, imInG, imInB):
def split(pilimIn, imOutR, imOutG, imOutB):
def superpose(imIn1, imIn2):
def tagOneColorPalette(value, color):
```

■ MODULE : **mambaComposed.contrasts**

This module provides a set of functions to perform morphological contrast operators (gradient, top-hat transform,...) using mamba. it works with imageMb instances as defined in mamba.

■ FUNCTIONS :

```
def blackTopHat(imIn, imOut, n, se=structuringElement([0, 1, 2, 3, 4, 5, 6], mamba.HEXAGONAL)):
def gradient(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
```

```
mamba.HEXAGONAL)):
def halfGradient(imIn, imOut, type='intern', n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6], mamba.HEXAGONAL)):
def regularisedGradient(imIn, imOut, n, grid=DEFAULT_GRID):
def supBlackTopHat(imIn, imOut, n, grid=DEFAULT_GRID):
def supWhiteTopHat(imIn, imOut, n, grid=DEFAULT_GRID):
def whiteTopHat(imIn, imOut, n, se=structuringElement([0, 1, 2, 3, 4, 5, 6], mamba.HEXAGONAL)):
```

■ MODULE : **mambaComposed.erodil**

This module provides a set of functions to perform erosions and dilations using Mamba base functions. It works with imageMb instances as defined in Mamba.

■ CLASS :

structuringElement

- ◆ `__cmp__(self, otherSE)`
- ◆ `__init__(self, directions, grid)`
- ◆ `__repr__(self)`
- ◆ `getDirections(self, withoutZero=False)`
- ◆ `getGrid(self)`
- ◆ `hasZero(self)`
- ◆ `rotate(self, step=1)`
- ◆ `setAs(self, se)`
- ◆ `transpose(self)`

■ FUNCTIONS :

```
def conjugateHexagonalDilate(imIn, imOut, size, edge=EMPTY):
```

```
def conjugateHexagonalErode(imIn, imOut, size, edge=FILLED):
def dilate(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6], mamba.HEXAGONAL), edge=EMPTY):
def dodecagonalDilate(imIn, imOut, size, edge=EMPTY):
def dodecagonalErode(imIn, imOut, size, edge=FILLED):
def doublePointDilate(imIn, imOut, d, n, grid=DEFAULT_GRID, edge=EMPTY):
def doublePointErode(imIn, imOut, d, n, grid=DEFAULT_GRID, edge=FILLED):
def erode(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6], mamba.HEXAGONAL), edge=FILLED):
def linearDilate(imIn, imOut, d, n=1, grid=DEFAULT_GRID, edge=EMPTY):
def linearErode(imIn, imOut, d, n=1, grid=DEFAULT_GRID, edge=FILLED):
def octogonalDilate(imIn, imOut, size, edge=EMPTY):
def octogonalErode(imIn, imOut, size, edge=FILLED):
```

MODULE : **mambaComposed.erodilLarge**

This module provides a set of functions performing erosions and dilations with large structuring elements. They are built with special shift operators written in C, together with special 'infFarNeighbor' and 'supFarNeighbor' functions.

FUNCTIONS :

```
def largeDodecagonalDilate(imIn, imOut, size, edge=EMPTY):
def largeDodecagonalErode(imIn, imOut, size, edge=FILLED):
def largeHexagonalDilate(imIn, imOut, size, edge=EMPTY):
```

```
def largeHexagonalErode(imIn, imOut, size, edge=FILLED):
def largeLinearDilate(imIn, imOut, dir, size, grid=DEFAULT_GRID,
edge=EMPTY):
def largeLinearErode(imIn, imOut, dir, size, grid=DEFAULT_GRID,
edge=FILLED):
def largeOctagonalDilate(imIn, imOut, size, edge=EMPTY):
def largeOctagonalErode(imIn, imOut, size, edge=FILLED):
def largeSquareDilate(imIn, imOut, size, edge=EMPTY):
def largeSquareErode(imIn, imOut, size, edge=FILLED):
```

MODULE : **mambaComposed.filter**

This module provides a set of functions to perform morphological filtering operations using mamba. it works with imageMb instances as defined in mamba.

FUNCTIONS :

```
def alternateFilter(imIn, imOut, n, openFirst, se=structuringElement([0, 1, 2,
3, 4, 5, 6], mamba.HEXAGONAL)):
```

```
def autoMedian(imIn, imOut, n, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
mamba.HEXAGONAL)):
def fullAlternateFilter(imIn, imOut, n, openFirst, se=structuringElement([0, 1,
2, 3, 4, 5, 6], mamba.HEXAGONAL)):
def linearAlternateFilter(imIn, imOut, n, openFirst, grid=DEFAULT_GRID):
def simpleLevelling(imIn, imMask, imOut, grid=DEFAULT_GRID):
def strongLevelling(imIn, imOut, n, eroFirst, grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.geodesy**

This module provides a set of functions to perform geodesic computations using Mamba based functions. It includes build and dualbuild operations, geodesic erosion and dilation, computation of maxima and minima... it works with imageMb instances as defined in mamba.

FUNCTIONS :

```
def build(imMask, imInout, grid=DEFAULT_GRID):
def closeHoles(imIn, imOut, grid=DEFAULT_GRID):
def dualBuild(imMask, imInout, grid=DEFAULT_GRID):
```

```
def geodesicDilate(imIn, imMask, imOut, n=1, se=structuringElement([0, 1, 2,
3, 4, 5, 6], mamba.HEXAGONAL)):
def geodesicDistance(imIn, imMask, imOut, se=structuringElement([0, 1, 2, 3,
4, 5, 6], mamba.HEXAGONAL)):
def geodesicErode(imIn, imMask, imOut, n=1, se=structuringElement([0, 1, 2,
3, 4, 5, 6], mamba.HEXAGONAL)):
def maxima(imIn, imOut, h=1, grid=DEFAULT_GRID):
def minima(imIn, imOut, h=1, grid=DEFAULT_GRID):
def removeEdgeParticles(imIn, imOut, grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.measure**

This module provides a set of functions which perform measure operations on mamba images. It works with imageMb instances as defined in mamba.

FUNCTIONS :

```
def computeArea(imIn, scale=(1.0, 1.0)):
```

```
def computeComponentsNumber(imIn, grid=DEFAULT_GRID):
def computeConnectivityNumber(imIn, grid=DEFAULT_GRID):
def computeDiameter(imIn, dir, scale=(1.0, 1.0), grid=DEFAULT_GRID):
def computeFeretDiameters(imIn, scale=(1.0, 1.0)):
def computePerimeter(imIn, scale=(1.0, 1.0), grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.miscellaneous**

This module regroups functions/operators that could not be regrouped with other operators because of their unique nature or other peculiarity. As such, it regroups some utility functions.

FUNCTIONS :

```
def ceilingAdd(imIn1, imIn2, imOut):
```

```
def ceilingAddConst(imIn, v, imOut):
def drawEdge(imOut, thick=1):
def floorSub(imIn1, imIn2, imOut):
def floorSubConst(imIn, v, imOut):
def isotropicDistance(imIn, imOut, edge=FILLED):
def translate(imIn, imOut, deltaX, deltaY, v=0):
```

MODULE : **mambaComposed.openclose**

This module provides a set of functions to perform opening and closing operations using mamba. it works with imageMb instances as defined in mamba. All the closing and opening operation defined in this module use erosion, dilation and build functions with user-defined edge settings. The functions define a default edge which can be changed (see the modules erodil and geodesy).

FUNCTIONS :

```
def buildClose(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
mamba.HEXAGONAL)):
```

```
def buildOpen(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
mamba.HEXAGONAL)):
def close(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
mamba.HEXAGONAL), edge=FILLED):
def infClose(imIn, imOut, n, grid=DEFAULT_GRID):
def linearClose(imIn, imOut, dir, n, grid=DEFAULT_GRID, edge=FILLED):
def linearOpen(imIn, imOut, dir, n, grid=DEFAULT_GRID, edge=FILLED):
def open(imIn, imOut, n=1, se=structuringElement([0, 1, 2, 3, 4, 5, 6],
mamba.HEXAGONAL), edge=FILLED):
def supOpen(imIn, imOut, n, grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.residues**

This module provides a set of functions to perform residual operations using mamba. It works with imageMb instances as defined in mamba. A residual transformation is built by subtracting two sequences of primitive operators to get residues and by computing the supremum of these residues. The position in the sequence where this maximum occurs is also computed (it is called associated function and is generally a 32-bit image). These residues are defined on binary and greytone images.

FUNCTIONS :

```
def binarySkeletonByOpening(imIn, imOut1, imOut2, grid=DEFAULT_GRID,
```

```
edge=FILLED):
def binaryUltimateErosion(imIn, imOut1, imOut2, grid=DEFAULT_GRID,
edge=FILLED):
def fullRegularisedGradient(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def quasiDistance(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def skeletonByOpening(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def ultimateBuildOpening(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def ultimateErosion(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def ultimateIsotropicOpening(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
def ultimateOpening(imIn, imOut1, imOut2, grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.segment**

This module provides a set of functions to perform segmentation operations using mamba. it works with imageMb instance as defined in mamba.

FUNCTIONS :

```
def fastSKIZ(imIn, imOut, grid=DEFAULT_GRID):
```

```
def geodesicSKIZ(imIn, imMask, imOut, grid=DEFAULT_GRID):
def markerControlledWatershed(imIn, imMarkers, imOut,
grid=DEFAULT_GRID):
def mosaic(imIn, imOut, imWts, grid=DEFAULT_GRID):
def mosaicGradient(imIn, imOut, grid=DEFAULT_GRID):
def valuedWatershed(imIn, imOut, grid=DEFAULT_GRID):
```

MODULE : **mambaComposed.sequence**

This module provides classes, methods and functions to perform morphological computations over sequences of images using mamba. This module can be considered as a restricted 3-D extension of mamba.

CLASS :

sequenceMb

- ◆ `__getitem__(self, key)`
- ◆ `__init__(self, *args, **kwargs)`
- ◆ `__iter__(self)`
- ◆ `fill(self, v)`
- ◆ `getDepth(self)`
- ◆ `getLength(self)`
- ◆ `getName(self)`
- ◆ `getSize(self)`
- ◆ `hideAllImages(self)`

- ◆ `hideImage(self, index)`
- ◆ `load(self, path, rgbfilter=None)`
- ◆ `next(self)`
- ◆ `reset(self)`
- ◆ `resetPalette(self)`
- ◆ `setPalette(self, pal)`
- ◆ `showAllImages(self)`
- ◆ `showImage(self, index)`

FUNCTIONS :

- def** `closeByCylinderSequence(sequence, height, section):`
- def** `copySequence(sequenceIn, sequenceOut):`
- def** `dilateByCylinderSequence(sequence, height, section):`
- def** `erodeByCylinderSequence(sequence, height, section):`
- def** `openByCylinderSequence(sequence, height, section):`

MODULE : **mambaComposed.statistic**

This module provides a set of functions to compute statistical values inside a mamba image.

FUNCTIONS :

- def** `getMean(imIn):`
- def** `getMedian(imIn):`
- def** `getVariance(imIn):`

MODULE : **mambaComposed.thinthick**

This module contains morphological Hit-or-Miss, thinning and thickening operators of the Mamba Image library, together with various homotopic and geodesic functions derived from these operators. They use imageMb image instances as defined in the mamba module.

CLASS :

doubleStructuringElement

- ◆ `__init__(self, *args)`
- ◆ `__repr__(self)`
- ◆ `flip(self)`
- ◆ `getCSE(self)`
- ◆ `getGrid(self)`
- ◆ `getStructuringElement(self, ground)`
- ◆ `rotate(self, step=1)`

FUNCTIONS :

- def** `binaryHMT(imIn, imOut, dse, edge=EMPTY):`
- def** `blackClip(imIn, imOut, step=0, grid=DEFAULT_GRID):`
- def** `computeSKIZ(imIn, imOut, grid=DEFAULT_GRID):`
- def** `endPoints(imIn, imOut, grid=DEFAULT_GRID, edge=FILLED):`
- def** `fullGeodesicThick(imIn, imMask, imOut, dse):`
- def** `fullGeodesicThin(imIn, imMask, imOut, dse):`

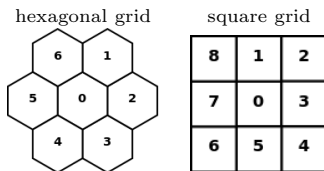
- def** `fullThick(imIn, imOut, dse):`
- def** `fullThin(imIn, imOut, dse, edge=EMPTY):`
- def** `geodesicThick(imIn, imMask, imOut, dse):`
- def** `geodesicThin(imIn, imMask, imOut, dse):`
- def** `homotopicReduction(imIn, imOut, grid=DEFAULT_GRID):`
- def** `infThin(imIn, imOut, dse, edge=EMPTY):`
- def** `multiplePoints(imIn, imOut, grid=DEFAULT_GRID):`
- def** `rotatingGeodesicThick(imIn, imMask, imOut, dse):`
- def** `rotatingGeodesicThin(imIn, imMask, imOut, dse):`
- def** `rotatingThick(imIn, imOut, dse):`
- def** `rotatingThin(imIn, imOut, dse, edge=FILLED):`
- def** `supThick(imIn, imOut, dse):`
- def** `thick(imIn, imOut, dse):`
- def** `thickD(imIn, imOut, grid=DEFAULT_GRID):`
- def** `thickL(imIn, imOut, grid=DEFAULT_GRID):`
- def** `thickM(imIn, imOut, grid=DEFAULT_GRID):`
- def** `thin(imIn, imOut, dse, edge=EMPTY):`
- def** `thinD(imIn, imOut, grid=DEFAULT_GRID, edge=EMPTY):`
- def** `thinL(imIn, imOut, grid=DEFAULT_GRID, edge=EMPTY):`
- def** `thinM(imIn, imOut, grid=DEFAULT_GRID, edge=EMPTY):`
- def** `whiteClip(imIn, imOut, step=0, grid=DEFAULT_GRID, edge=FILLED):`

GENERAL

GRID AND EDGE:

Mamba can work with two grids : **HEXAGONAL** and **SQUARE**. The **DEFAULT_GRID** is used by the function when no other grid is specified. Its value, **HEXAGONAL** at start, can be changed with the appropriate function. Two edge behaviors are defined : **EMPTY** and **FILLED**.

DIRECTIONS/NEIGHBORS :



STRUCTURING ELEMENTS :

List of structuring elements defined in Mamba

