Working with images in Python

FONDAZIONE BRUNO KESSLER

Marco Mina
Daniele Morabito

Presentation outline

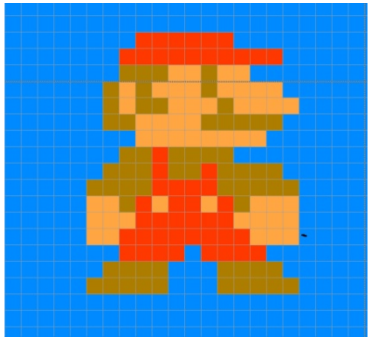
- PIL
- Open/Save images in Python
- Convert images between different formats
- Simple pixel-level manipulations
- Image processing libraries: Mahotas & Scikit-Image

PIL- Python Image Library

- Create, load, modify and convert image files
- Simple filtering and enhancement algorithms
- Supports all common image formats (JPEG, BMP, TIFF, ...)

https://pypi.python.org/pypi/Pillow/2.0.0

Step 1: load/visualize/save an image

Generic code	Example
from PIL import Image	from PIL import Image
original=Image.open("filename")	original=Image.open("test.jpg")
original.show()	original.show()
original.save("destination","filetype")	original.save("mario","png")
	

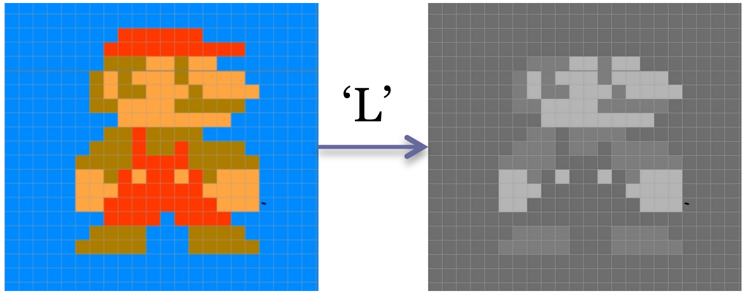
Step 2: format conversion

Generic code

```
from PIL import Image
original=Image.open("filename")

converted_img=original.convert('L')

converted_img.show()
converted_img.save("destination","filetype")
```



The *mode* of an image describes the way it represents colors. Each mode is represented by a string:

Mode	Bands	Description
"1"	1	Black and white (monochrome), one bit per pixel.
"L"	1	Gray scale, one 8-bit byte per pixel.
"P"	1	Palette encoding: one byte per pixel, with a palette of class `ImagePalette` translating the pixels to colors. This mode is experimental; refer to the online documentation[6].
"RGB"	3	True red-green-blue color, three bytes per pixel.
"RGBA"	4	True color with a transparency band, four bytes per pixel, with the A channel varying from 0 for transparent to 255 for opaque.
"CMYK"	4	Cyan-magenta-yellow-black color, four bytes per pixel.
"YCbCr"	3	Color video format, three 8-bit pixels.
"I"	1	32-bit integer pixels.
"F"	1	32-bit float pixels.

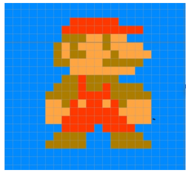
Step 3: import/export to ndarray

Generic code

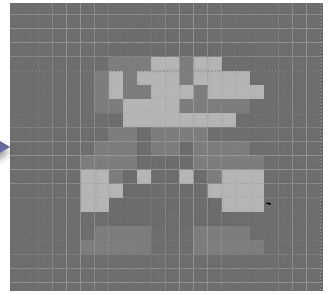
```
from PIL import Image
original=Image.open("filename")
converted_img=original.convert('L')

import numpy as np
ndarr=np.array(converted_img)
imported_img=Image.fromarray(ndarr)

imported_img.show()
imported_img.save("destination","filetype")
```



'L'



np.array

Image.fromarray

```
array([[133, 105, 114, ..., 110, 114, 108],
       [193, 110, 107, ..., 110, 103, 113],
       [131, 109, 110, ..., 111, 115, 106],
       ...,
       [123, 113, 112, ..., 114, 107, 110],
       [202, 114, 107, ..., 108, 114, 113],
       [127, 107, 108, ..., 108, 114, 106]], dtype=uint8)
```

Step 4: low-level pixel manipulation

Simple manipulation

```
from PIL import Image
original=Image.open("filename")
converted_img=original.convert('L')
import numpy as np
ndarr=np.array(converted_img)

ndarr[1:5,1:5]=0

imported_img=Image.fromarray(ndarr)
imported_img.show()
imported_img.save("destination","filetype")
```

Selecting and manipulating pixels below a given threshold

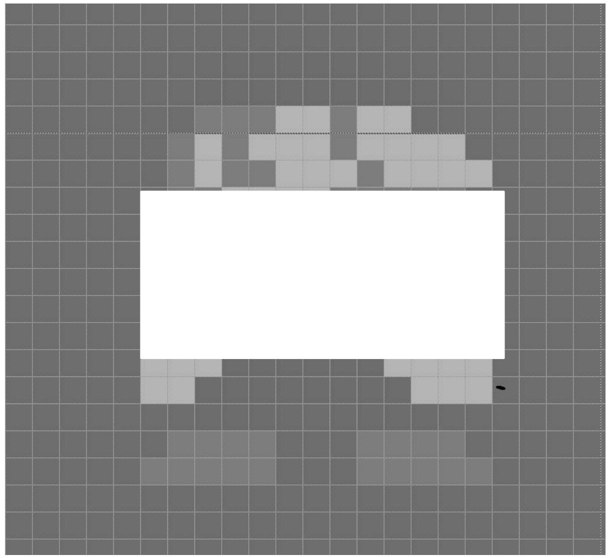
```
from PIL import Image
original=Image.open("filename")
converted_img=original.convert('L')
import numpy as np
ndarr=np.array(converted_img)

threshold=150
ndarr[ndarr<threshold]=0

imported_img=Image.fromarray(ndarr)
imported_img.show()
imported_img.save("destination","filetype")
```

Exercise 1

- Download an image from Internet and load it
- Convert it to gray scale
- Set to 0 all pixels with intensity greater than 150
- Visualize or save the result
- Set to 255 all pixels within a rectangle of arbitrary size at the center of the image



Tip: converting an array of booleans into an image

The trick: convert booleans to integers

from PIL import Image
original=Image.open(“filename”)
converted_img=original.convert(‘L’)
import numpy as np
ndarr=np.array(converted_img)

ndarr=ndarr<100

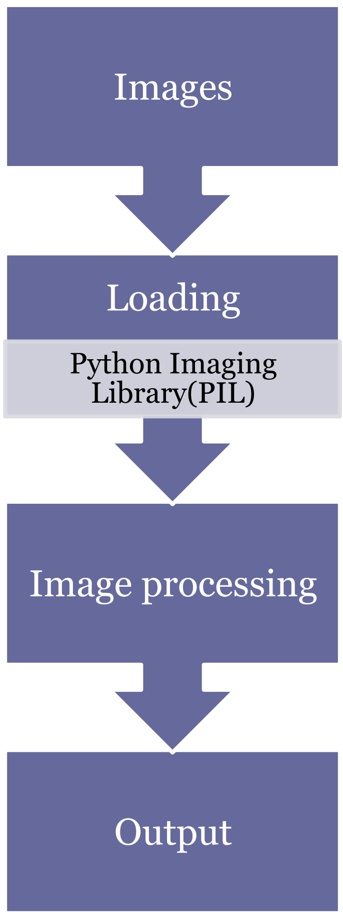
~~**imported_img=Image.fromarray(ndarr)**~~
ndarr=ndarr.astype(np.uint8)

imported_img=Image.fromarray(ndarr)
imported_img.show()
imported_img.save(“destination”,”filetype”)

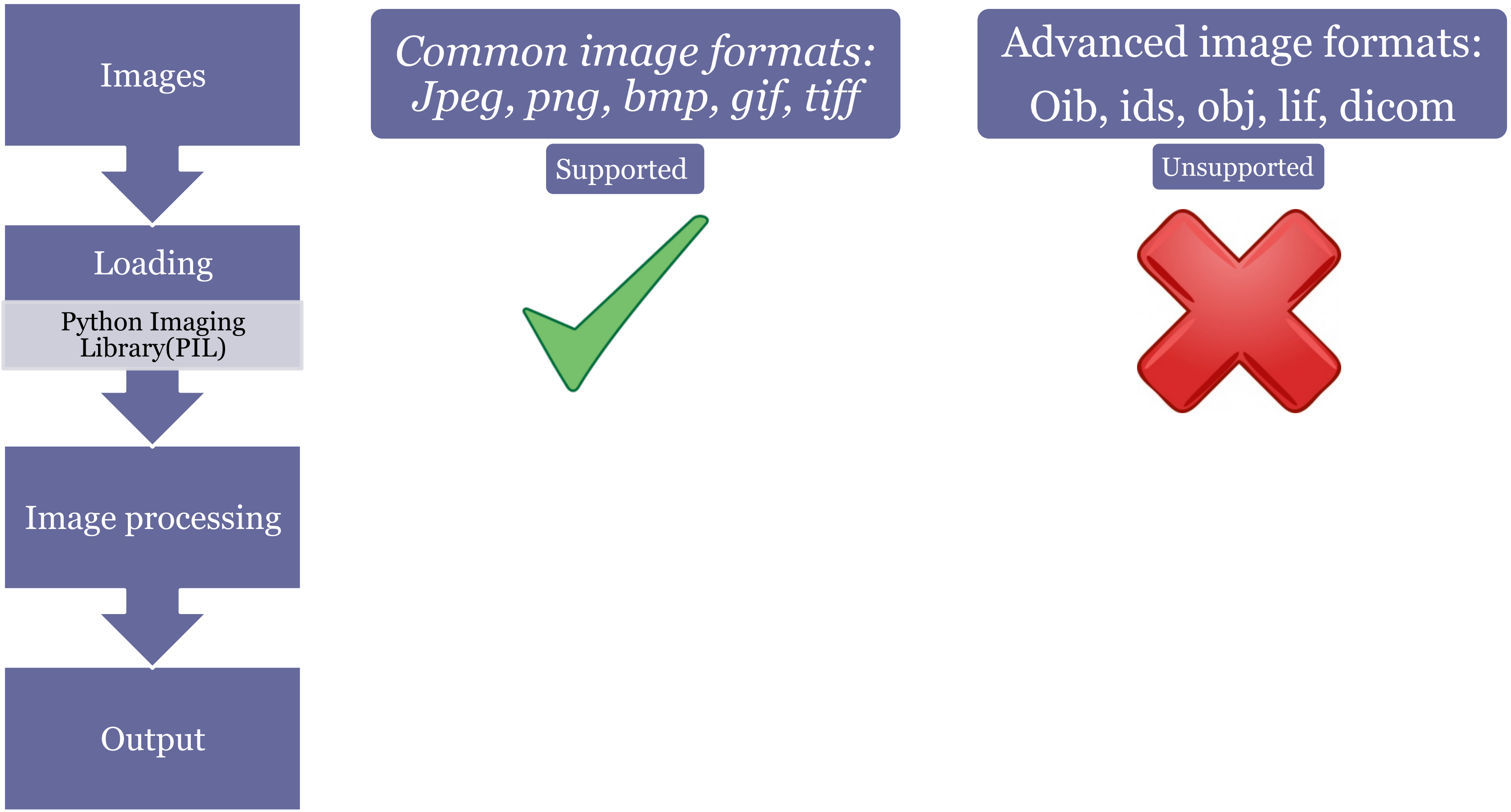
Image processing and computer vision in Python

Marco Mina
Daniele Morabito
FONDAZIONE BRUNO KESSLER

Typical image processing workflow



Formats directly supported by PIL



Importing formats not directly supported

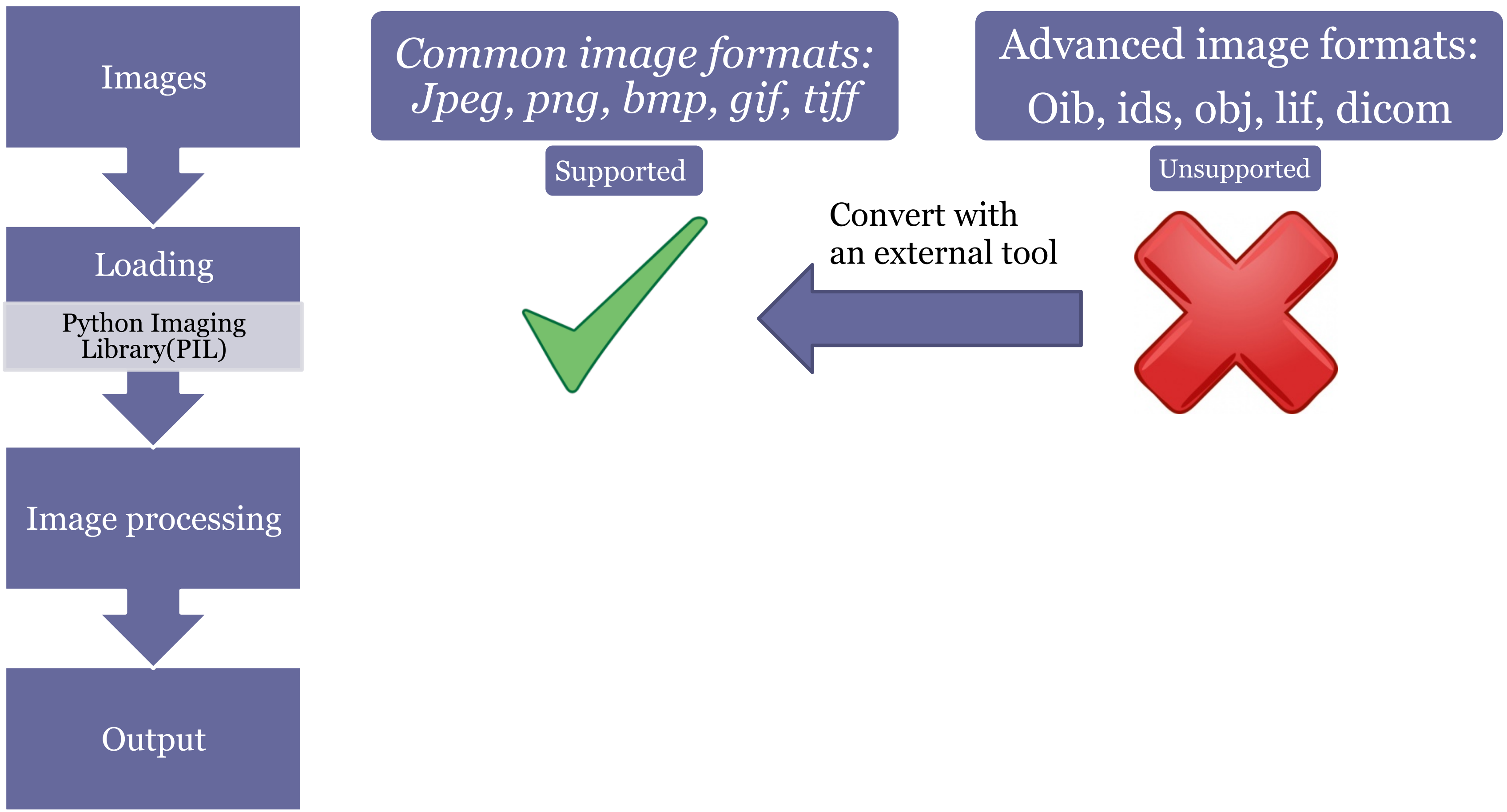
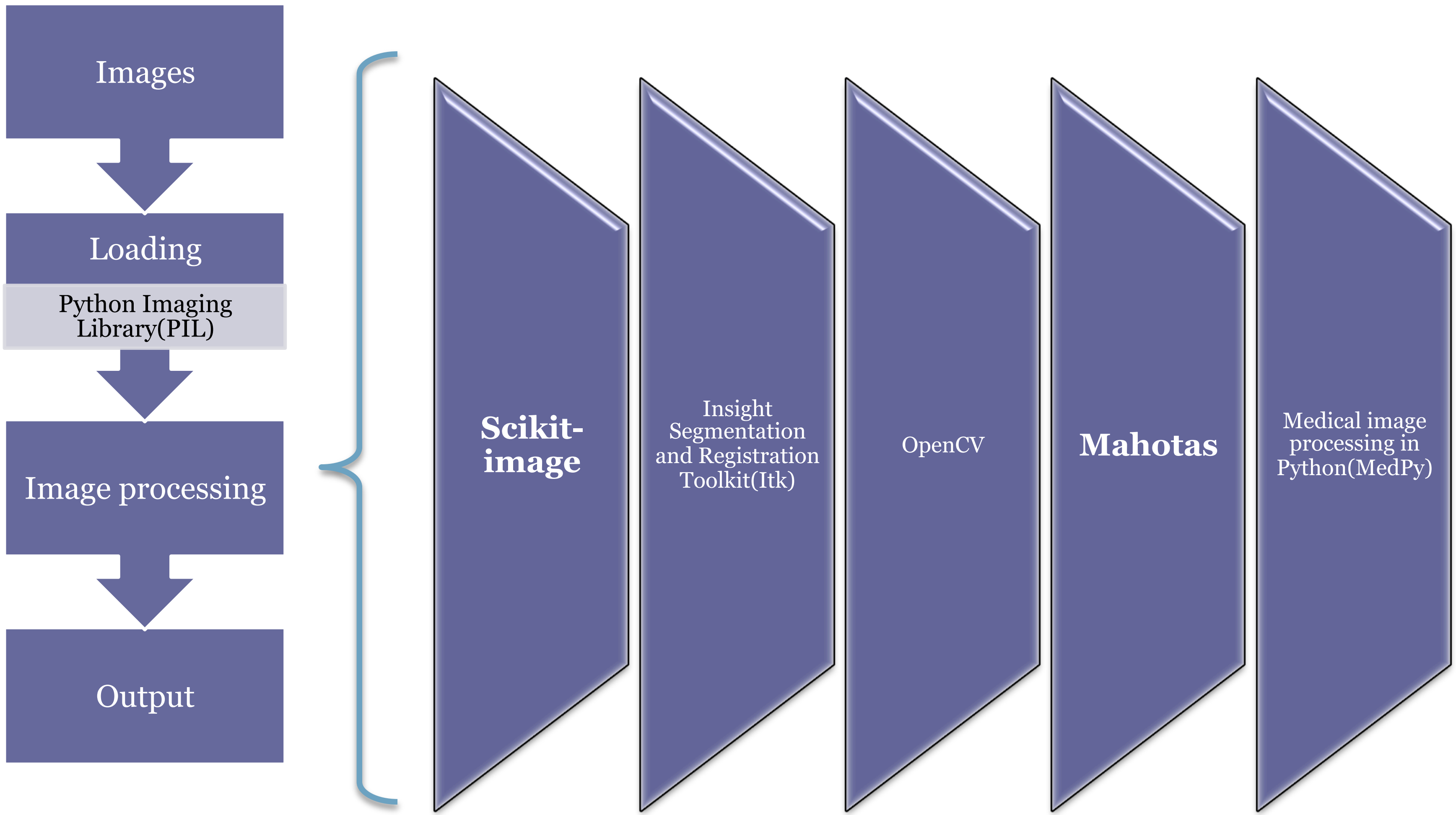


Image processing libraries



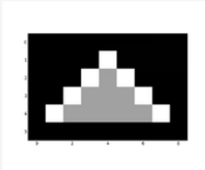
scikit-image
image processing in python

Home Download Gallery Documentation Source

Image processing in Python

scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.

Download

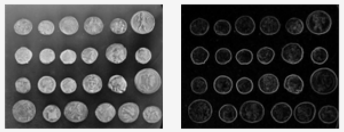


Getting Started

Filtering an image with `scikit-image` is easy! For more examples, please visit our gallery.

```
from skimage import data, io, filter

image = data.coins() # or any NumPy array!
edges = filter.sobel(image)
io.imshow(edges)
```



Mahotas

Python Computer Vision Library

This library of fast computer vision algorithms (all implemented in C++) operates over numpy arrays for convenience.

Notable algorithms:

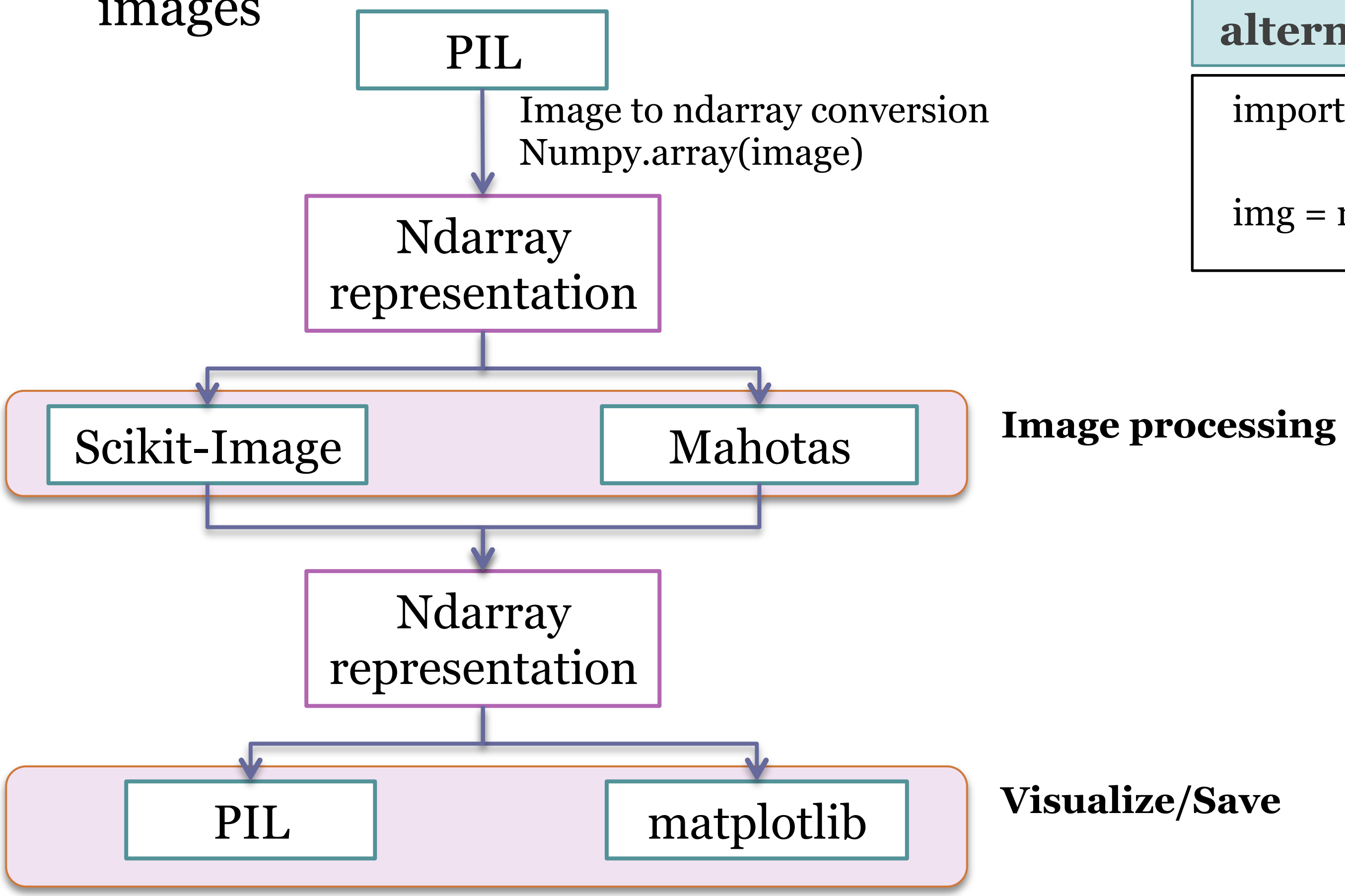
- watershed.
- convex points calculations.
- hit & miss. thinning.
- Zernike & Haralick, LBP, and TAS features.
- freeimage based numpy image loading (requires freeimage libraries to be installed).
- Speeded-Up Robust Features (SURF), a form of local features.
- thresholding.
- convolution.
- Sobel edge detection.
- spline interpolation

http://scikit-image.org/

http://pythonhosted.org/mahotas/api.html

From PIL to Scikit/Mahotas

- Both libraries can work on ndarray representation of images



Loading images: an alternative for mahotas

```
import mahotas

img = mahotas.imread('test.jpg‘)
```

Matplotlib: visualizing ndarray images

Generic code

```
from PIL import Image
import numpy as np

original=Image.open("filename")
converted_img=original.convert('L')
ndarr=np.array(converted_img)

import pyplot
pylab.imshow(tmp)
pylab.show()
```

Directly plot ndarrays

No need to convert back to PIL images

Can plot any type of image (even booleans)

Convolutions with scipy

```
from PIL import Image
import numpy as np

original = Image.open('colored.tif')
bw = original.convert('L')
bw.show()

import scipy.ndimage

tmp = numpy.array(bw)

kernel = np.array([[1.0/9,1.0/9,1.0/9],[1.0/9,1.0/9,1.0/9],[1.0/9,1.0/9,1.0/9]])

tmp = scipy.ndimage.filters.convolve(tmp, kernel)

tmpimage = Image.fromarray(tmp)

tmpimage.show()
```

Median filter with Scikit-image

```
from PIL import Image
import numpy as np

original = Image.open('easy.tif')
bw = original.convert('L')
bw.show()
tmp = numpy.array(bw)


from skimage import filter

processed_ndarr = filter.median_filter(tmp) # median filter

temp = Image.fromarray(processed_ndarr)

temp.show()
```

Otsu with Mahotas

```
from PIL import Image
import numpy as np

original = Image.open('colored.tif')
bw = original.convert('L')
bwarray = numpy.array(bw)

import mahotas

threshold = mahotas.otsu(bwarray)

bwarray[bwarray<threshold] = 0

bwarray[bwarray>=threshold] = 255

img_bw_thresholded = Image.fromarray(bwarray)
img_bw_thresholded.show()
bw.show()
```