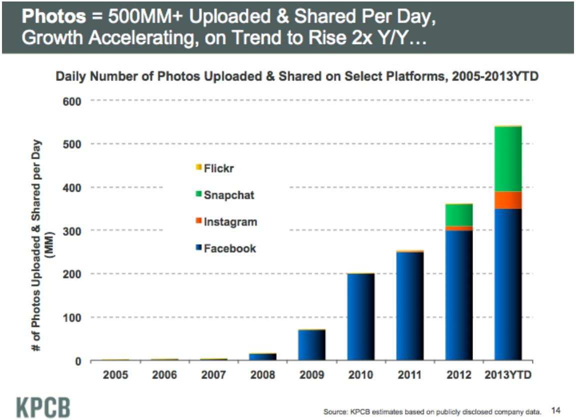
Image processing with `scikit-image`

Emmanuelle Gouillart
Surface, Glass and Interfaces, CNRS/Saint-Gobain

Paris-Saclay Center for Data Science

The world is getting more and more visual

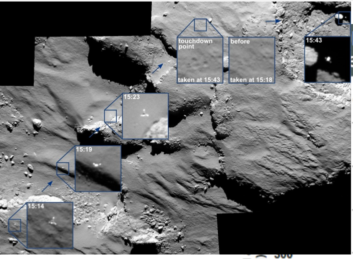


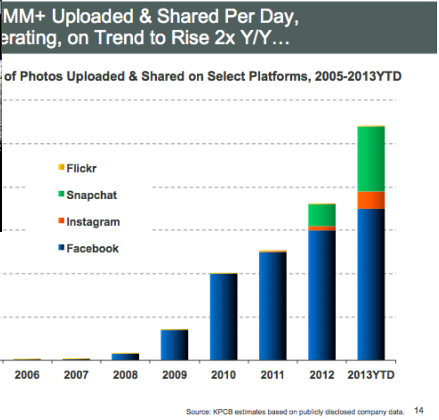
Photos = 500MM+ Uploaded & Shared Per Day,
Growth Accelerating, on Trend to Rise 2x Y/Y…

Daily Number of Photos Uploaded & Shared on Select Platforms, 2005-2013YTD

Source: KPCB estimates based on publicly disclosed company data. 14

The world is getting more and more visual





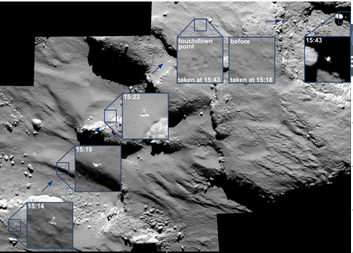
MM+ Uploaded & Shared Per Day,
erating, on Trend to Rise 2x Y/Y…

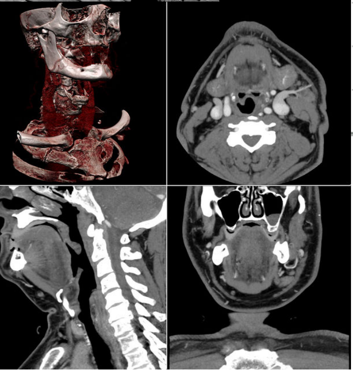
of Photos Uploaded & Shared on Select Platforms, 2005-2013YTD

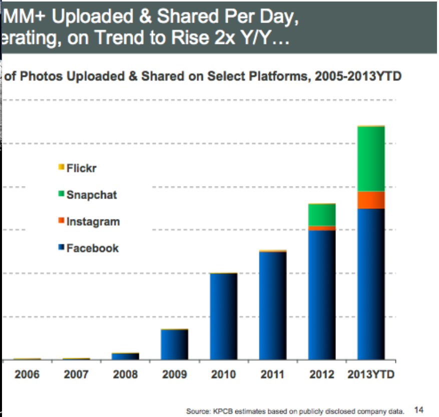
KPCB

Source: KPCB estimates based on publicly disclosed company data.

The world is getting more and more visual





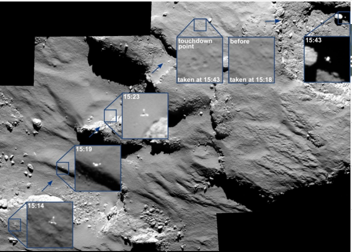


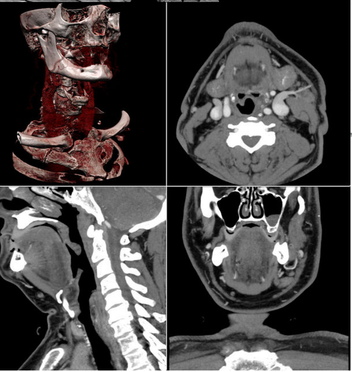
MM+ Uploaded & Shared Per Day,
erating, on Trend to Rise 2x Y/Y…

of Photos Uploaded & Shared on Select Platforms, 2005-2013YTD

Source: KPCB estimates based on publicly disclosed company data. 14

The world is getting more and more visual

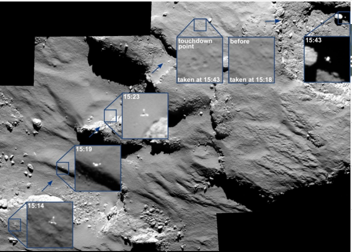


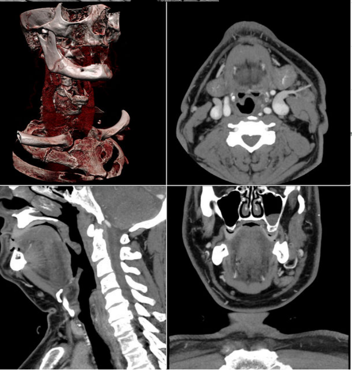


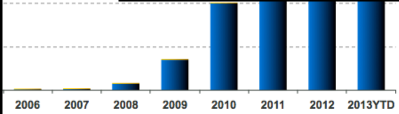
MM+ Uploade
erating, on Tre
of Photos Uploade
Flickr
Snapchat
Instagram
Facebook
2006
2007
2008
2009
2010
2011
2012
2013YTD
Russian → English
PARKING SERVICE VEHICLES
Russian → English
СТОЯНКА СЛУЖЕБНОГО АВТОТРАНСПОРТА

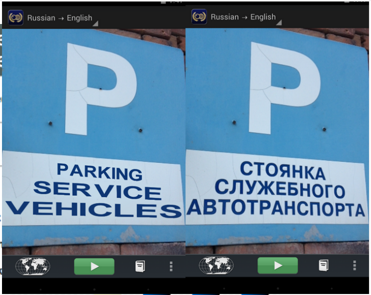
Source: KPCB estimates based on publicly disclosed company data. 14

The world is getting more and more visual

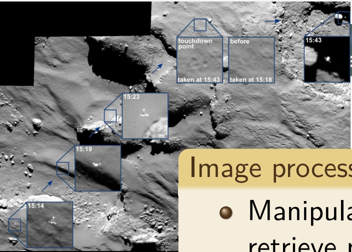


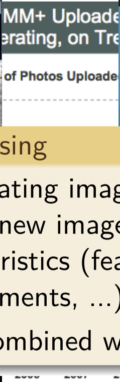






The world is getting more and more visual





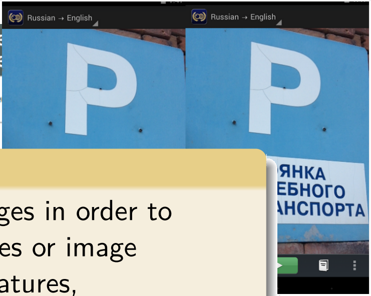
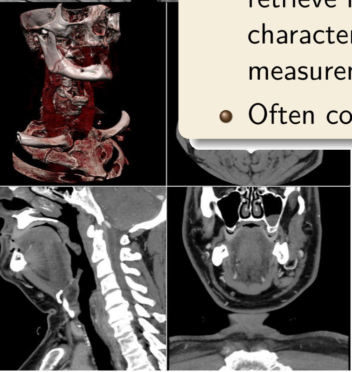


Image processing

- Manipulating images in order to retrieve new images or image characteristics (features, measurements, ...)
- Often combined with machine learning



scikit-image

http://scikit-image.org/

A module of the Scientific Python stack

- Language: Python
    - Core modules: NumPy, SciPy, matplotlib
        - Application modules: scikit-learn, scikit-image, pandas, ...

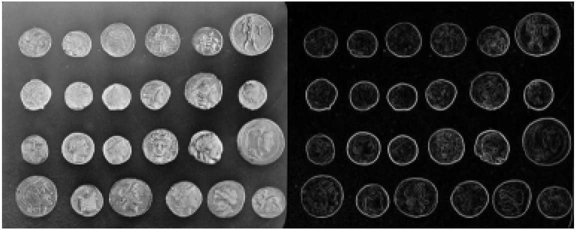
A general-purpose image processing library

- open-source (BSD)
- not an application (ImageJ)
- less specialized than other libraries (e.g. OpenCV for computer vision)

1 Principle

1 First steps

```python
from skimage import data, io, filter

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

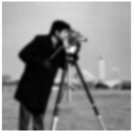


My environment: IPython interpreter + text editor
Ipython notebook nice for demos/trial and error

1 Manipulating images as numpy arrays

- numpy arrays as arguments and outputs

```
>>> from skimage import io, filter
>>> camera_array = io.imread('camera_image.png')
>>> type(camera_array)
<type 'numpy.ndarray'>
>>> camera_array.dtype
dtype('uint8')
>>> filtered_array = filter.gaussian_filter(
    camera_array, sigma=5)
>>> type(filtered_array)
<type 'numpy.ndarray'>
>>> filtered_array.dtype
dtype('float64')
```

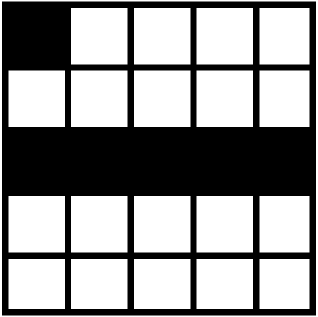


http://www.numpy.org/
scipy-lectures.github.io/intro/numpy/index.html

1 Manipulating images as numerical (numpy) arrays

- Pixels are arrays elements

```
import numpy as np
image = np.ones((5, 5))
image[0, 0] = 0
image[2, :] = 0
```

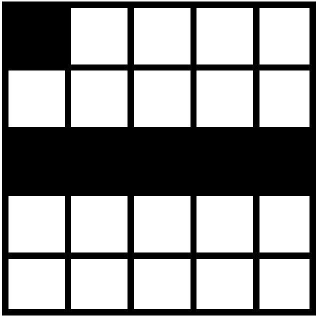


(use matplotlib for visualization: matplotlib.pyplot.imshow)

1 Manipulating images as numerical (numpy) arrays

- Pixels are arrays elements

```
import numpy as np
image = np.ones((5, 5))
image[0, 0] = 0
image[2, :] = 0
```



(use matplotlib for visualization: matplotlib.pyplot.imshow)

1 Some magics inside

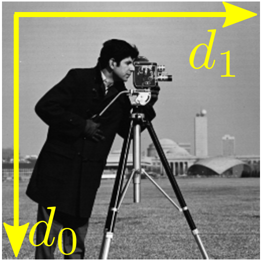
Don't let yourself be tricked by integer / float conversion!

```
>>> from skimage import data, filter
>>> camera_array = data.camera()
>>> camera_array.dtype
dtype('uint8')
>>> filtered_array = filter.gaussian_filter(
    camera_array, sigma=5)
>>> filtered_array.dtype
dtype('float64')
>>> camera_array.min(), camera_array.max()
(0, 255)
>>> filtered_array.min(), filtered_array.max()
(0.031287384322526979, 0.85609948978467 72)
```

1 An API relying mostly on functions

```
skimage.filter.gaussian_filter(image, sigma, output=
    None, mode='nearest', cval=0, multichannel=None)

Multi-dimensional Gaussian filter

Parameters
----------

image : array-like
    input image (grayscale or color) to filter.
sigma : scalar or sequence of scalars
    standard deviation for Gaussian kernel. The
        standard
    deviations of the Gaussian filter are given for
        each axis as a
    sequence, or as a single number, in which case it
        is equal for
    all axes.
output : array, optional
    The ``output`` parameter passes an array in which
        to store the
    filter output.
```

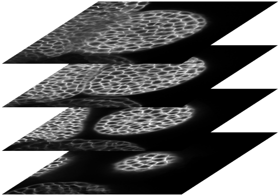
1 Images and dimensions



```
>>> data.camera().shape
(512, 512)
```

- Most functions suitable for 2-D gray- or color-scale images
- Some functions work with 3D images as well
- Check out `scipy.ndimage` for n-d functionnality.

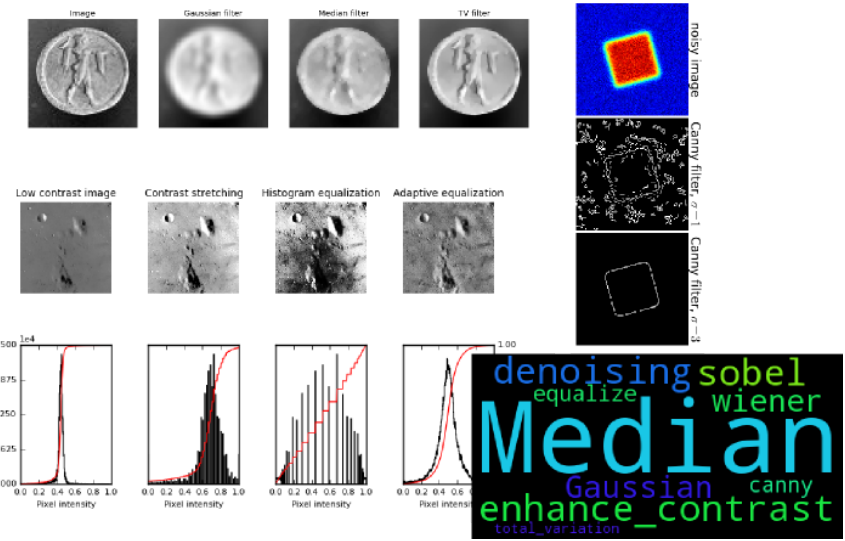
```
>>> coffee.shape
(400, 600, 3)
>>> red_channel =
coffee[..., 0]
```



(d_0, d_1, d_2)

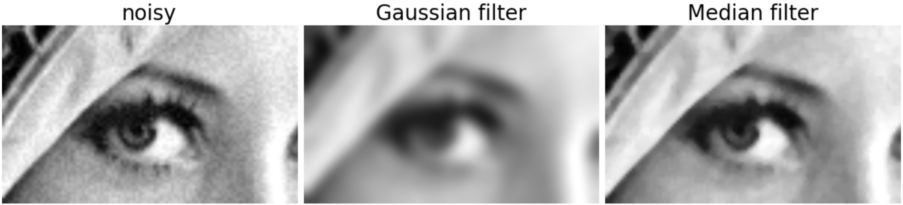
2 Some features

2 Filtering: transforming image data



skimage.filter, skimage.exposure, skimage.restoration

2 From very simple/classical algorithms



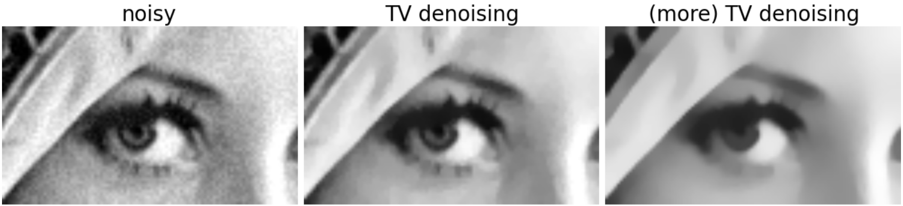
noisy

Gaussian filter

Median filter

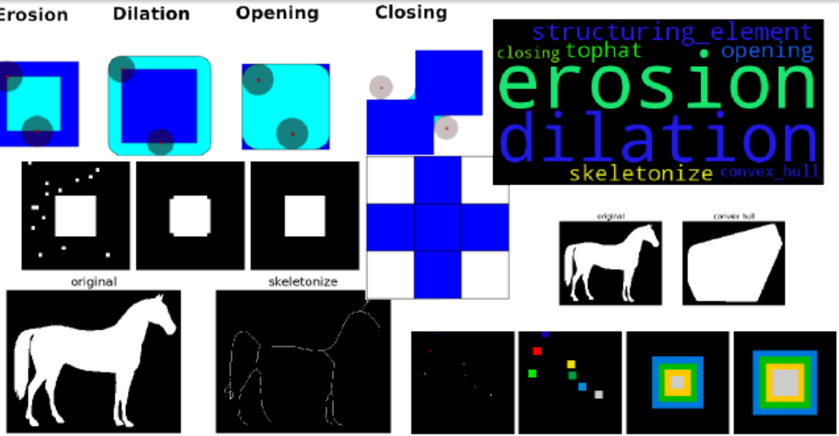
```
from skimage import data, filter, color
from skimage.morphology import disk

I = data.lena()
I = color.rgb2grey(I)
I = I[230:290, 220:320]

noisy = I + 0.4 * I.std() * np.random.random(I.shape)

gaussian_denoised = filter.gaussian_filter(noisy,
    sigma=2)
median_denoised = filter.rank.median(noisy, disk(3))
```

2 To more advanced/recent algorithms



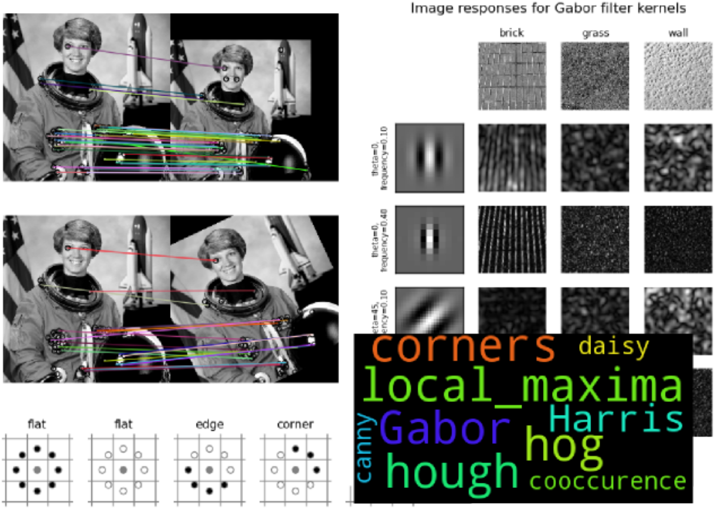
```
from skimage.filter import tv_denoise
from skimage import data

l = data.lena()
l = l[230:290, 220:320]
noisy = l + 0.4*l.std()*np.random.random(l.shape)

tv_denoised = tv_denoise(noisy, weight=10)
```

2 Mathematical morphology



skimage.morphology

2 Extracting features



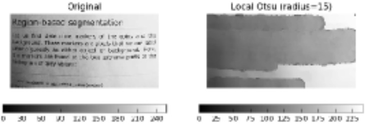
skimage.feature, skimage.filter

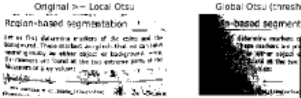
2 Geometrical transformations

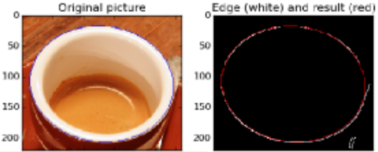
`skimage.transform`

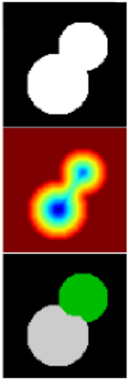
scale, zoom, rotate, swirl, warp, ...

2 Segmentation: labelling regions









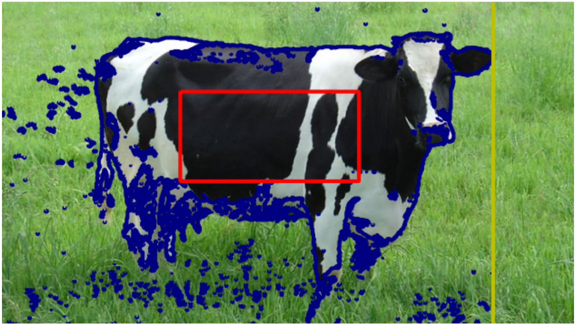
Felzenszwalbs's method

`skimage.segmentation`

2 Feature extraction followed by classification

Combining `scikit-image` and `scikit-learn`

- Extract features (`skimage.feature`)
  - Pixels intensity values (R, G, B)
  - Local gradients
  - More advanced descriptors: HOGs, Gabor, ...
- Train classifier with known regions
  - here, random forest classifier
- Classify pixels



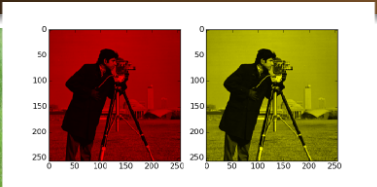
2 Measures on images

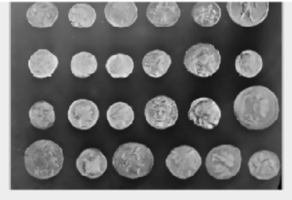
size
label
measure
histogram
regionprops
0
50
100
150
200
250
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300

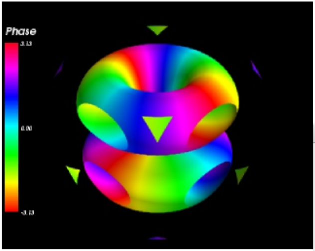
skimage.measure

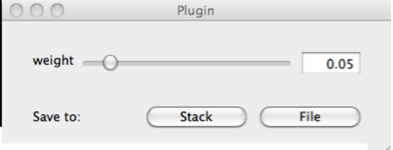
2 Visualizing images and more

matplotlib, mayavi

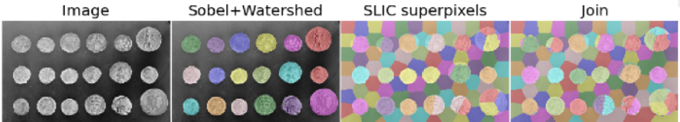








weight 0.05



Image

Sobel+Watershed

SLIC superpixels

Join

2 Development of `scikit-image`

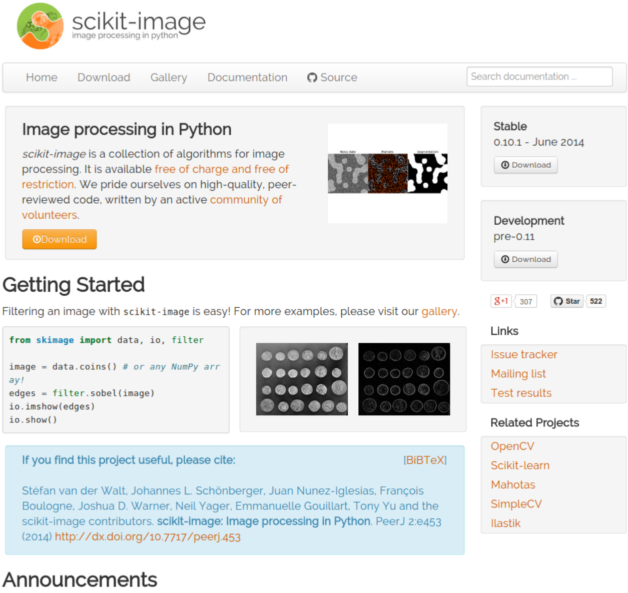
- Mature algorithms
- Only `Python` + `Cython` code for easier maintainability
- Hosted on GitHub
- Thorough code review by others: readability, PEP8, efficiency, ...
- 1-2 releases per year
- Core team of 5 persons (+ GSoc students)

2 Getting started: installing `scikit-image`

http://scikit-image.org/docs/dev/install.html

- Packaged on Ubuntu/Debian
- Shipped with all major Scientific Python distributions: Enthought Canopy, Anaconda, Python(x,y)

2 Getting started: finding documentation



scikit-image
image processing in python

Home | Download | Gallery | Documentation | Source | Search documentation ...

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

Stable
0.10.1 - June 2014
Download

Development
pre-0.11
Download

8+1 307 | Star 522

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 arr
ay!
edges = filter.sobel(image)
io.imshow(edges)
io.show()
```

If you find this project useful, please cite: [BiBTeX]

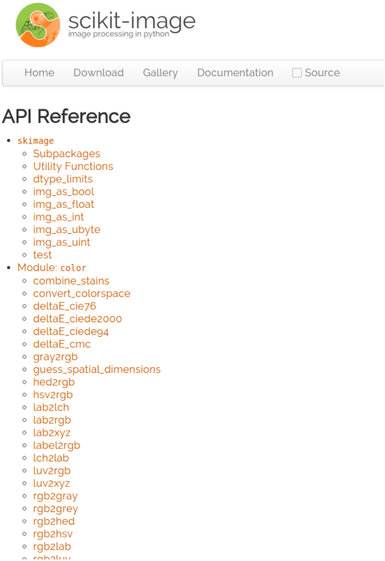
Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. **scikit-image: Image processing in Python**. PeerJ 2:e453 (2014) http://dx.doi.org/10.7717/peerj.453

Links
- Issue tracker
- Mailing list
- Test results

Related Projects
- OpenCV
- Scikit-learn
- Mahotas
- SimpleCV
- Ilastik

Announcements

2 Getting started: finding documentation

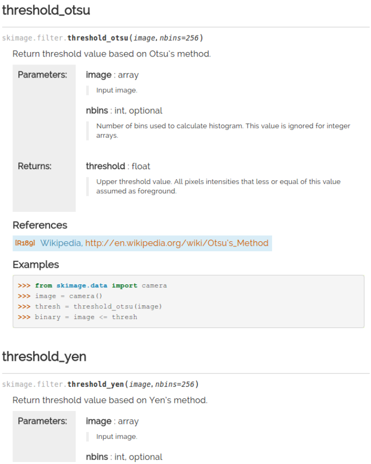


scikit-image
image processing in python

Home Download Gallery Documentation □ Source

API Reference

- skimage
  - Subpackages
  - Utility Functions
  - dtype_limits
  - img_as_bool
  - img_as_float
  - img_as_int
  - img_as_ubyte
  - img_as_uint
  - test
- Module: color
  - combine_stains
  - convert_colorspace
  - deltaE_cie76
  - deltaE_ciede2000
  - deltaE_ciede94
  - deltaE_cmc
  - gray2rgb
  - guess_spatial_dimensions
  - hed2rgb
  - hsv2rgb
  - lab2lch
  - lab2rgb
  - lab2xyz
  - label2rgb
  - lch2lab
  - luv2rgb
  - luv2xyz
  - rgb2gray
  - rgb2grey
  - rgb2hed
  - rgb2hsv
  - rgb2lab
  - rgb2luv



threshold_otsu

`skimage.filter.threshold_otsu(image, nbins=256)`

Return threshold value based on Otsu's method.

Parameters:	**image** : array Input image. **nbins** : int, optional Number of bins used to calculate histogram. This value is ignored for integer arrays.
Returns:	**threshold** : float Upper threshold value. All pixels intensities that less or equal of this value assumed as foreground.

References

[R189] Wikipedia, http://en.wikipedia.org/wiki/Otsu's_Method

Examples

```
>>> from skimage.data import camera
>>> image = camera()
>>> thresh = threshold_otsu(image)
>>> binary = image <= thresh
```

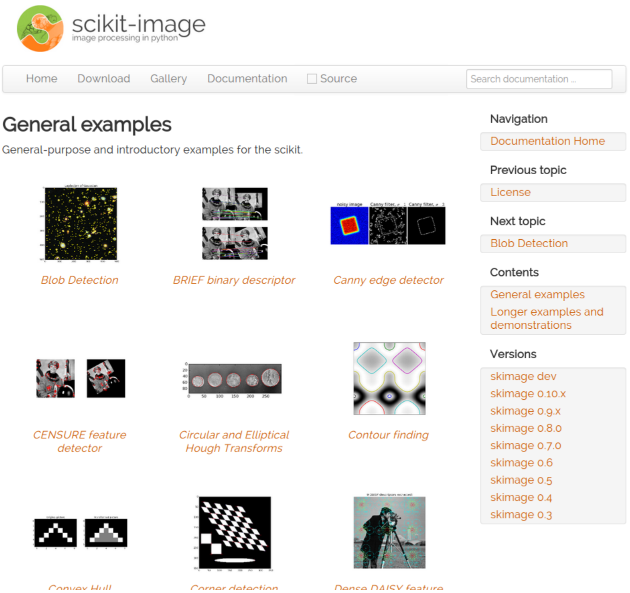
threshold_yen

`skimage.filter.threshold_yen(image, nbins=256)`

Return threshold value based on Yen's method.

Parameters:	**image** : array Input image. **nbins** : int, optional

2 Gallery of examples



scikit-image
image processing in python

Home Download Gallery Documentation □ Source Search documentation ...

General examples

General-purpose and introductory examples for the scikit.

Blob Detection

BRIEF binary descriptor

Canny edge detector

CENSURE feature detector

Circular and Elliptical Hough Transforms

Contour finding

Convex Hull

Corner detection

Dense DAISY feature

Navigation

Documentation Home

Previous topic

License

Next topic

Blob Detection

Contents

General examples

Longer examples and demonstrations

Versions

skimage dev

skimage 0.10.x

skimage 0.9.x

skimage 0.8.0

skimage 0.7.0

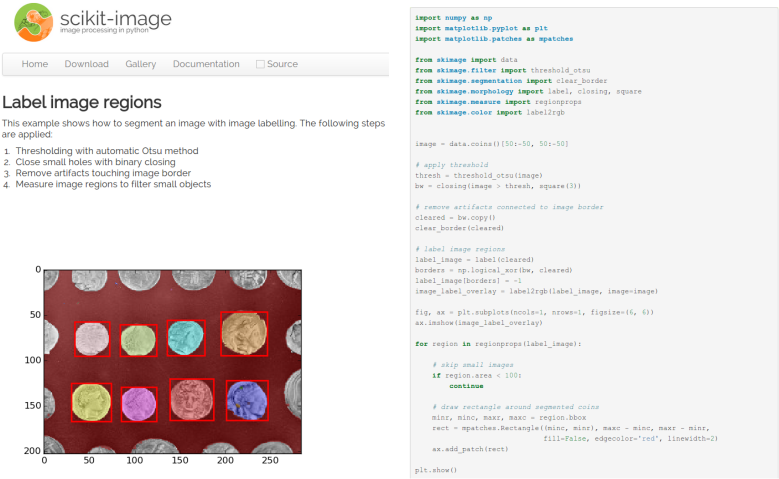
skimage 0.6

skimage 0.5

skimage 0.4

skimage 0.3

2 Getting started: finding documentation



Label image regions

This example shows how to segment an image with image labelling. The following steps are applied:

1. Thresholding with automatic Otsu method
2. Close small holes with binary closing
3. Remove artifacts touching image border
4. Measure image regions to filter small objects

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

from skimage import data
from skimage.filter import threshold_otsu
from skimage.segmentation import clear_border
from skimage.morphology import label, closing, square
from skimage.measure import regionprops
from skimage.color import label2rgb


image = data.coins()[50:-50, 50:-50]

# apply threshold
thresh = threshold_otsu(image)
bw = closing(image > thresh, square(3))

# remove artifacts connected to image border
cleared = bw.copy()
clear_border(cleared)

# label image regions
label_image = label(cleared)
borders = np.logical_xor(bw, cleared)
label_image[borders] = -1
image_label_overlay = label2rgb(label_image, image=image)

fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(6, 6))
ax.imshow(image_label_overlay)

for region in regionprops(label_image):

    # skip small images
    if region.area < 100:
        continue

    # draw rectangle around segmented coins
    minr, minc, maxr, maxc = region.bbox
    rect = mpatches.Rectangle((minc, minr), maxc - minc, maxr - minr,
                              fill=False, edgecolor='red', linewidth=2)
    ax.add_patch(rect)

plt.show()
```

2 Conclusions

scikit-image

- An image processing Python module relying on NumPy arrays
- Trade-off between performance and usability
- More and more features
- Try it out!
  - http://scikit-image.org/
  - https://www.youtube.com/watch?v=SE7h0IWD93Y (and others)
  - http://scipy-lectures.github.io/packages/scikit-image/