Image processing with `scikit-image`

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

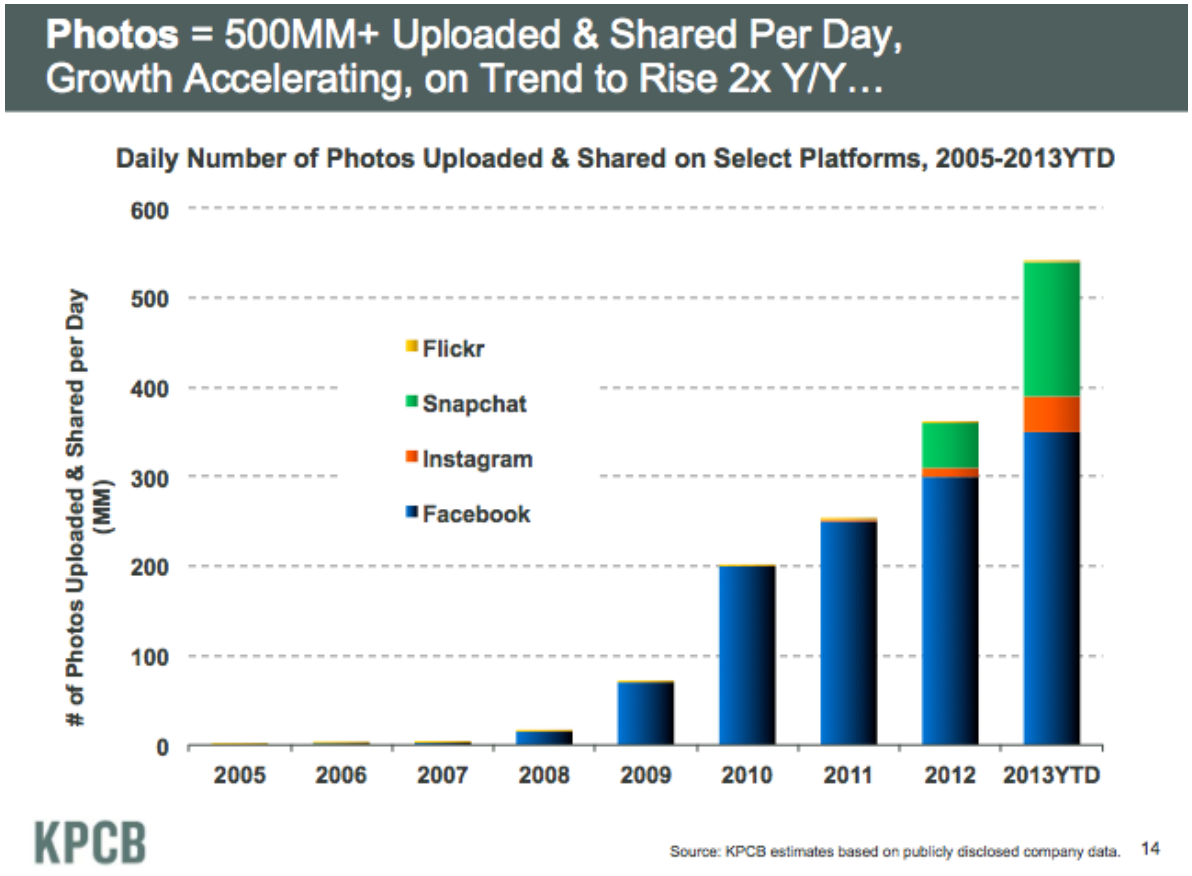
@EGouillart

scikit-image
image processing in python

The revolution of images

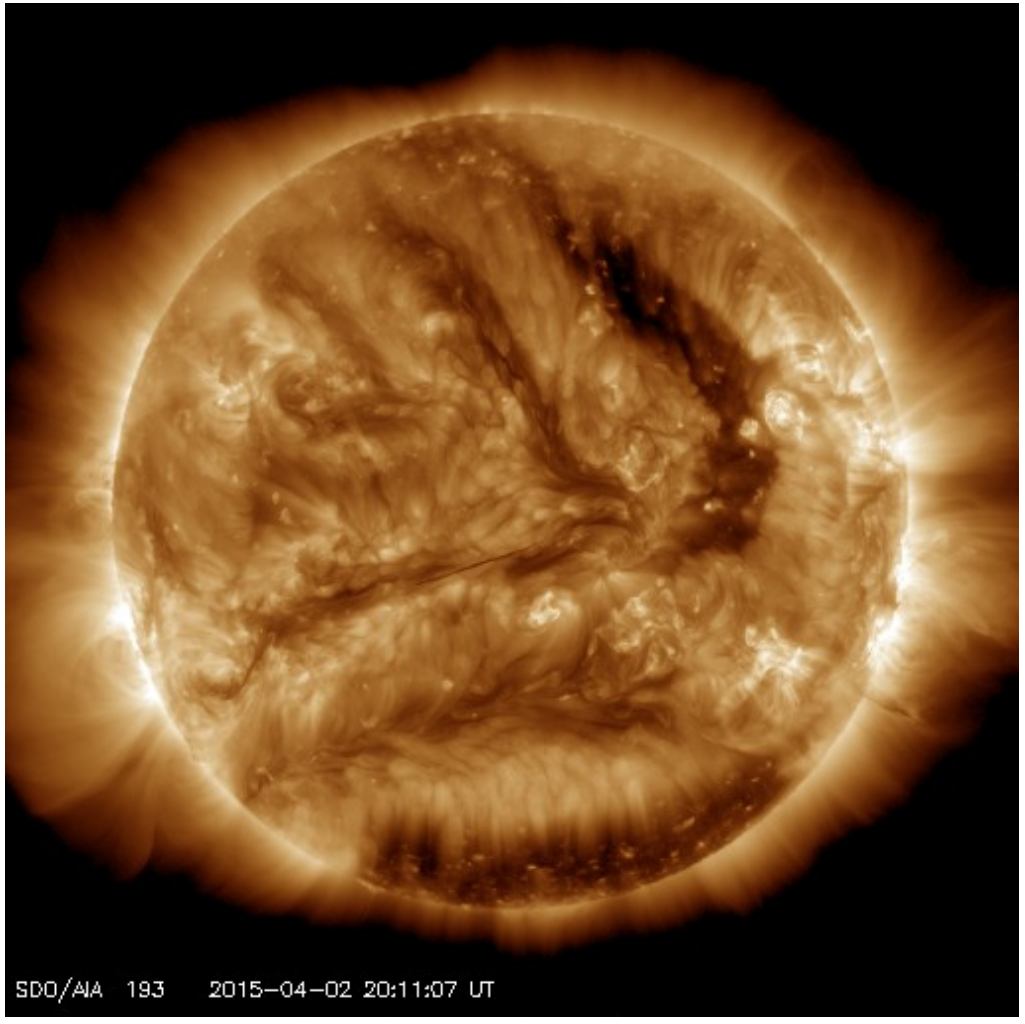
The revolution of images

A flood of images



several 10^8 images uploaded on Facebook each day

A flood of images



hundreds of terabytes of scientific data for scientific experiment

http://sdo.gsfc.nasa.gov/

A flood of images

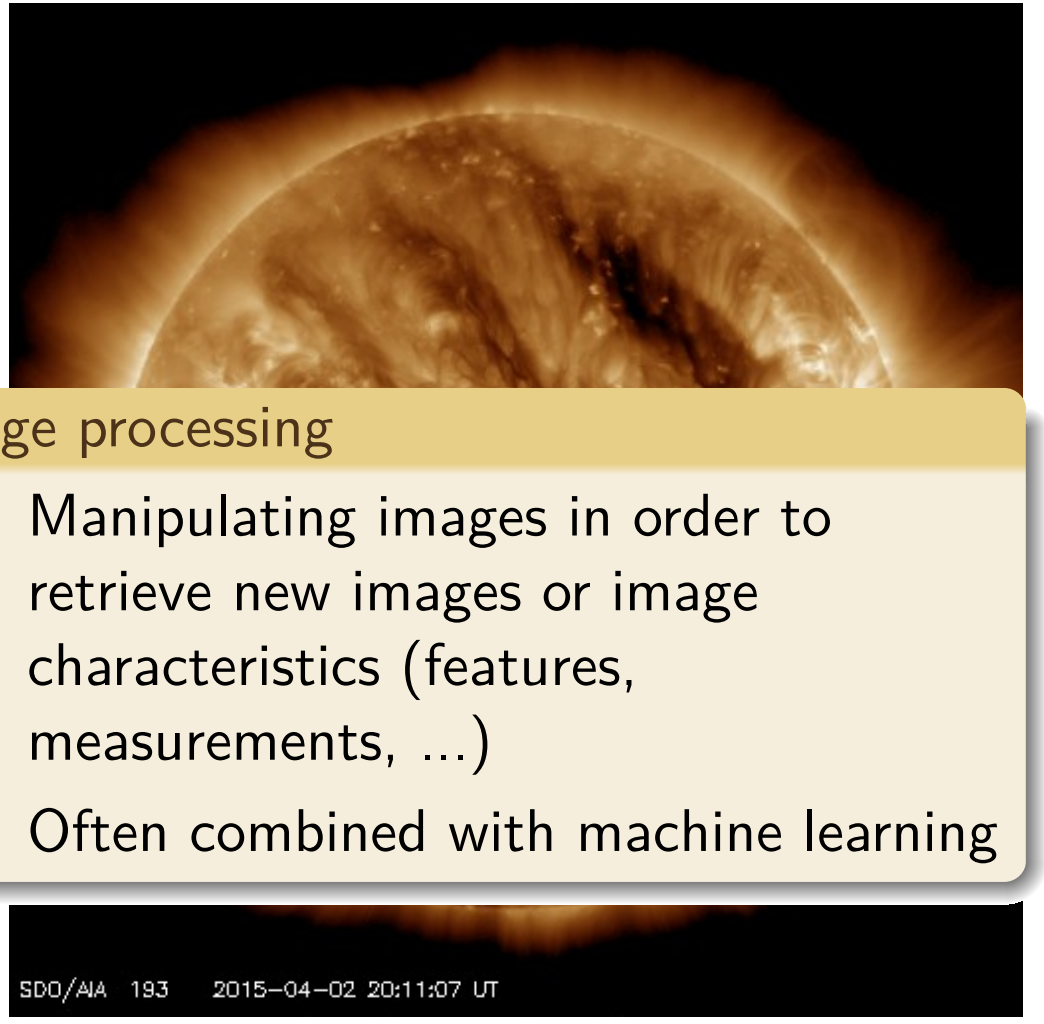


Image processing

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

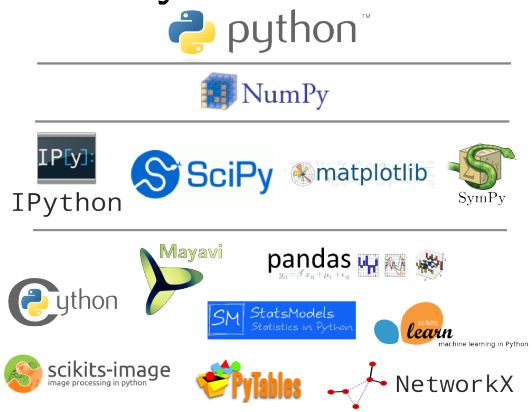
hundreds of terabytes of scientific data for scientific experiment

http://sdo.gsfc.nasa.gov/

Why use scikit-image?

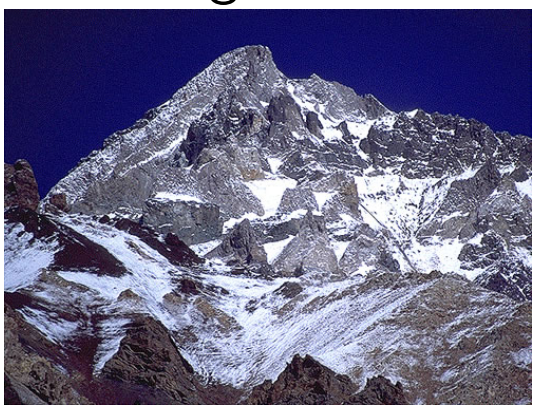
Lots of excellent tools available: OpenCV (computer vision), ImageJ (GUI-based software), ITK, ...

- Native NumPy compatibility

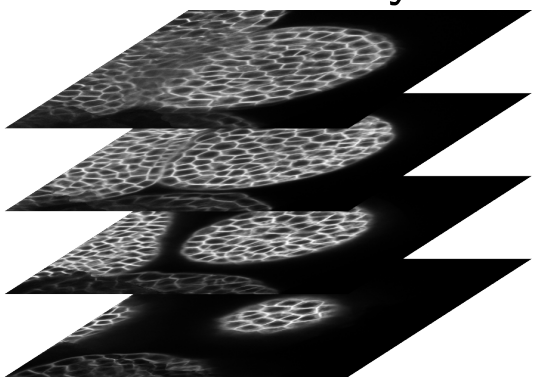


- Variety of features

- Gentle learning curve



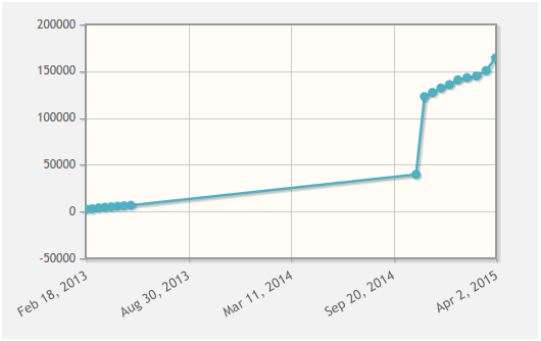
- Versatile - 3D-friendly



Datasheet

Package statistics

- `http://scikit-image.org/`
- Release 0.11 (1 - 2 release per year)
- Available in most Scientific Python Distributions: Canopy, Anaconda, ...
- Packaged on Ubuntu/Debian
- Among 1000 best ranked packages on PyPi



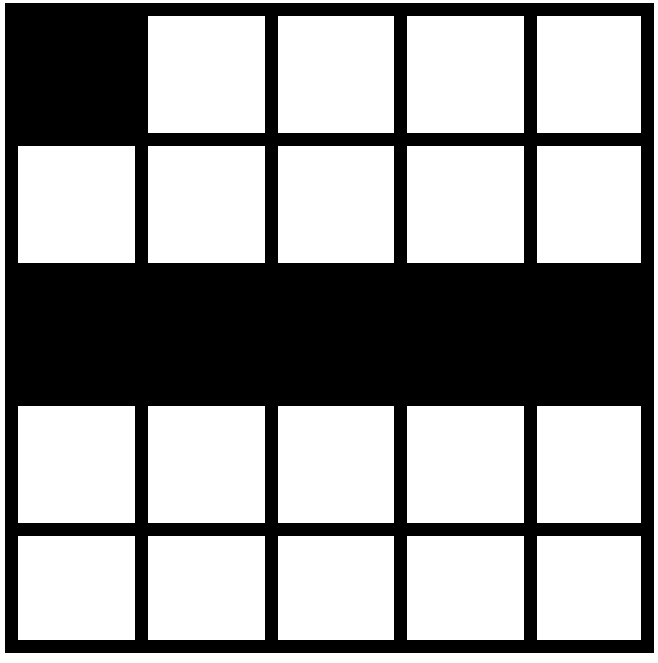
Development model

- Mature algorithms
- Only `Python` + `Cython` code for easier maintainability
- Thorough code review by others: readability, PEP8, efficiency, ...
- Core team of $5 - 10$ persons (close to applications)

Manipulating images as numerical (numpy) arrays

- Pixels are arrays elements

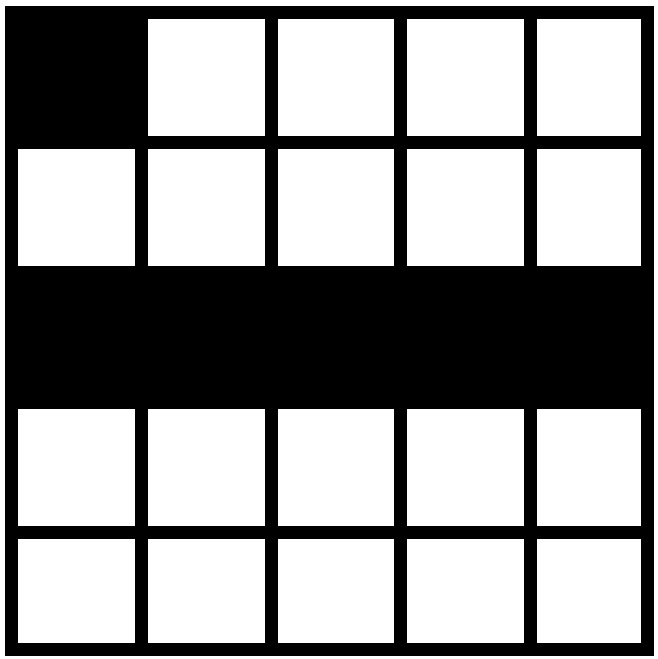
```
import numpy as np
image = np.ones((5, 5))
image[0, 0] = 0
image[2, :] = 0
```



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
```



```
>>> coffee.shape
(400, 600, 3)
>>> red_channel =
coffee[..., 0]
>>> image_3d =
np.ones((100, 100, 100))
```

NumPy-native: images as NumPy arrays

- NumPy arrays as arguments and outputs

```
>>> from skimage import io, filters
>>> camera_array = io.imread('camera_image.png')
>>> type(camera_array)
<type 'numpy.ndarray'>
>>> camera_array.dtype
dtype('uint8')
>>> filtered_array = filters.gaussian_filter(
    camera_array, sigma=5)
>>> type(filtered_array)
<type 'numpy.ndarray'>
>>> import matplotlib.pyplot as plt
>>> plt.imshow(filtered_array, cmap='gray')
```

An API relying mostly on functions

```
skimage.filters.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.
```

Getting started: finding documentation

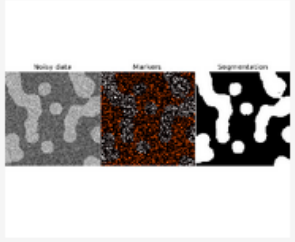
scikit-image
image processing in python

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



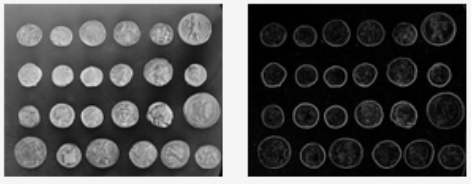
Stable
0.10.1 - June 2014
Download

Development
pre-0.11
Download

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()
```



+1 307 Star 522

Links
- Issue tracker
- Mailing list
- Test results

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

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

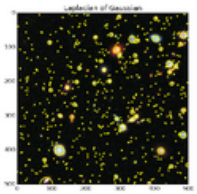
Gallery of examples

scikit-image
image processing in python

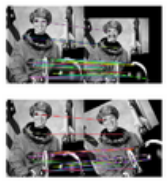
Home Download Gallery Documentation Source Search documentation ...

General examples

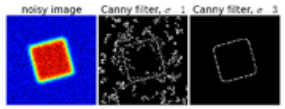
General-purpose and introductory examples for the scikit.



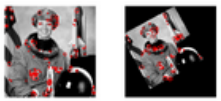
Blob Detection



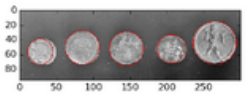
BRIEF binary descriptor



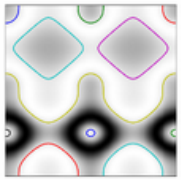
Canny edge detector



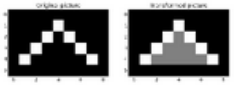
CENSURE feature detector



Circular and Elliptical Hough Transforms



Contour finding



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

Getting started: finding documentation

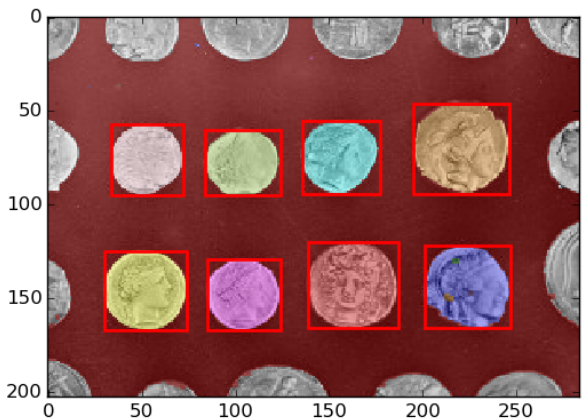
scikit-image
image processing in python

Home Download Gallery Documentation Source

Label image regions

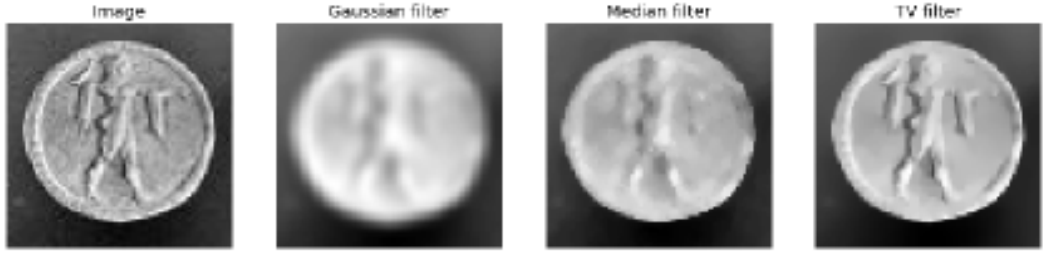
This example shows how to segment an image with image labelling. The following steps are applied:

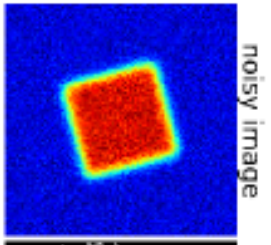
1. Thresholding with automatic Otsu method
2. Close small holes with binary closing
3. Remove artifacts touching image border
4. Measure image regions to filter small objects

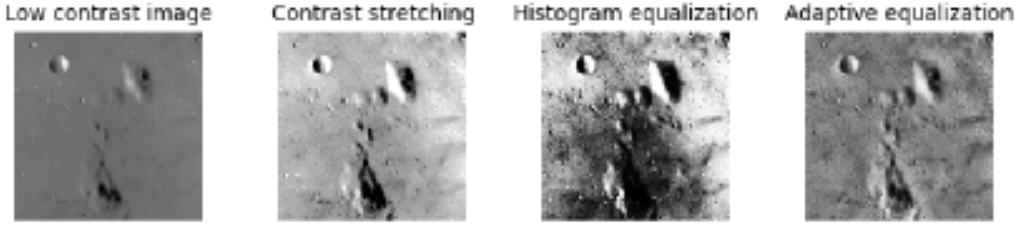


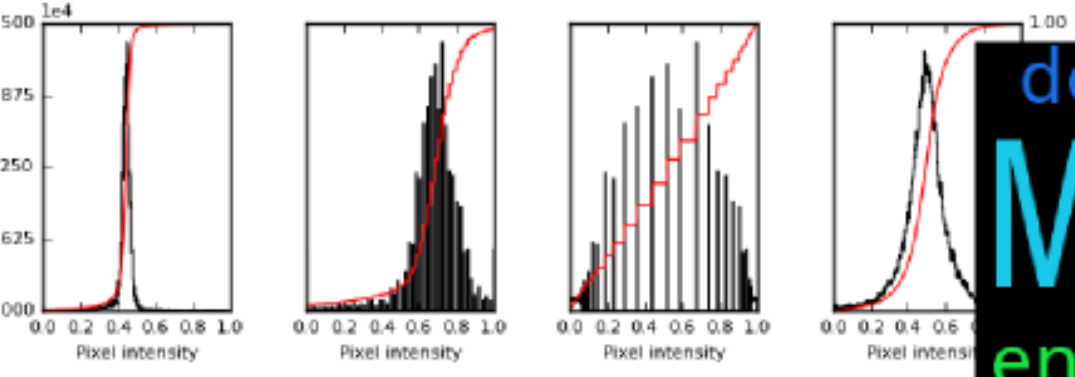

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

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()
```

Filtering: transforming image data



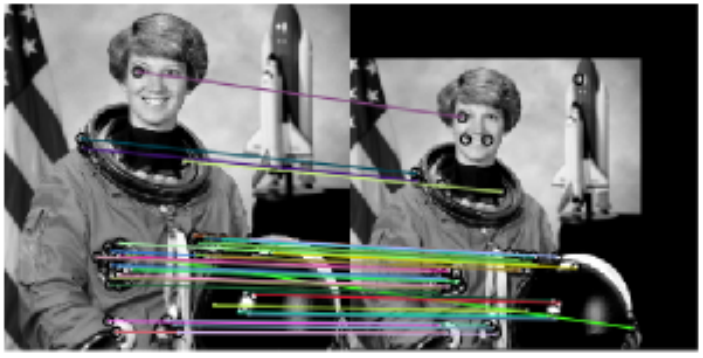


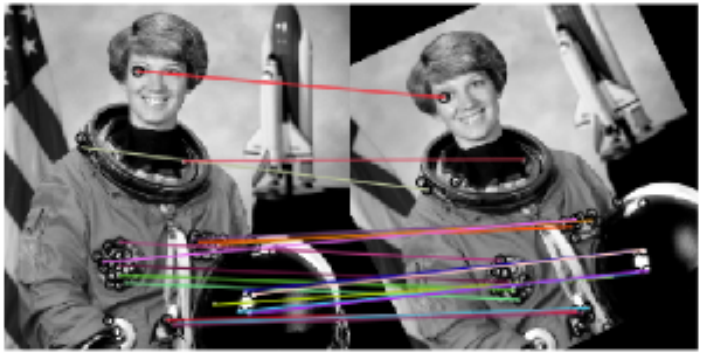


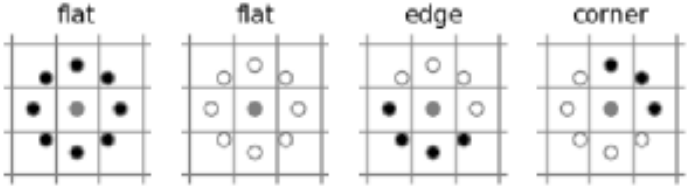


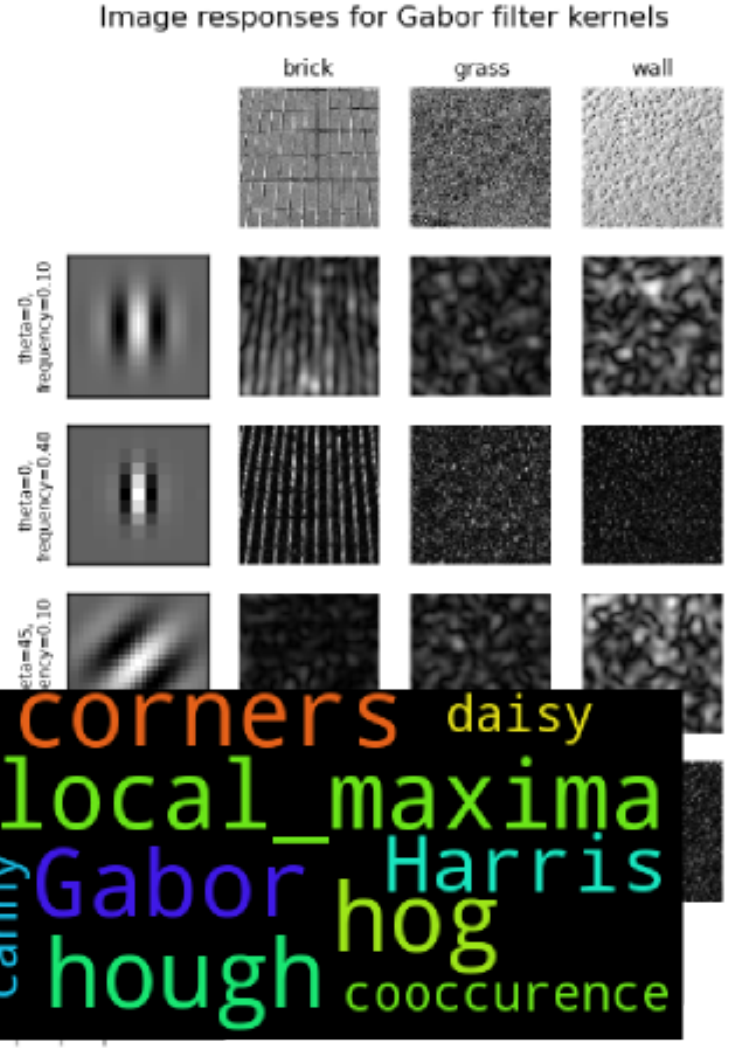
`skimage.filter`, `skimage.exposure`, `skimage.restoration`

Extracting features









`skimage.feature`, `skimage.filter`

Feature extraction followed by classification

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

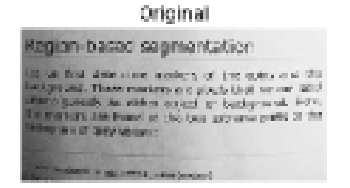
- Extract features (`skimage.feature`)
  - Pixels intensity values (R, G, B)
  - Local gradients
  - More advanced descriptors: HOGs, Gabor, ...
- Train classifier with known regions
  - here, random forest classifier
- Classify pixels

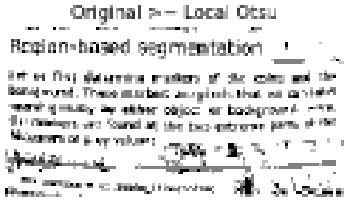
Geometrical transformations

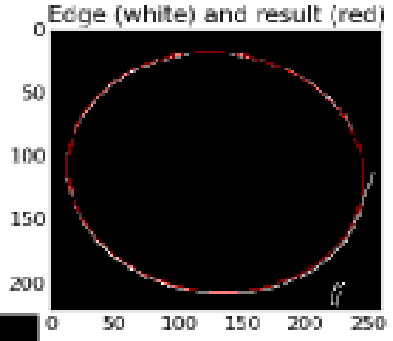
`skimage.transform`

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

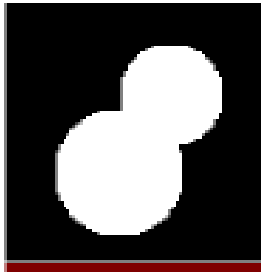
Segmentation: labelling regions

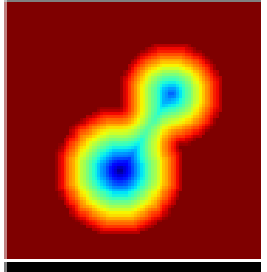


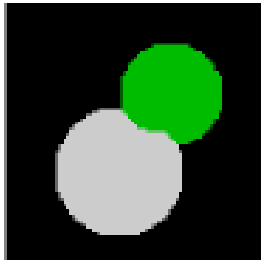




watershed
superpixel
thresholding
otsu
randomwalker





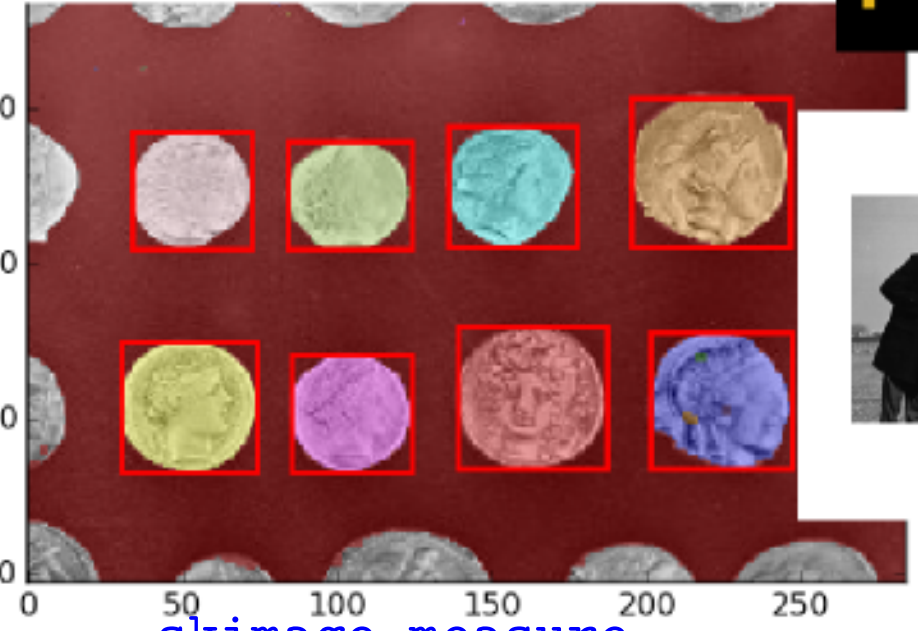


Felzenszwalbs's method

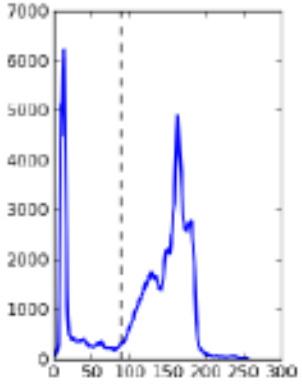
skimage.segmentation

Measures on images

size label
measure
histogram
regionprops

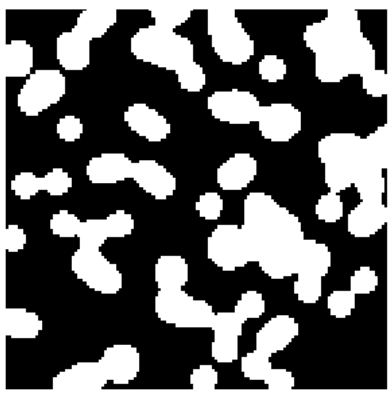


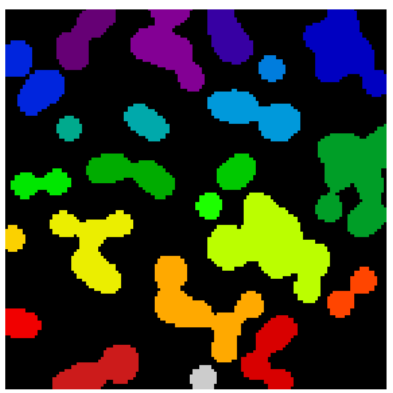
skimage.measure

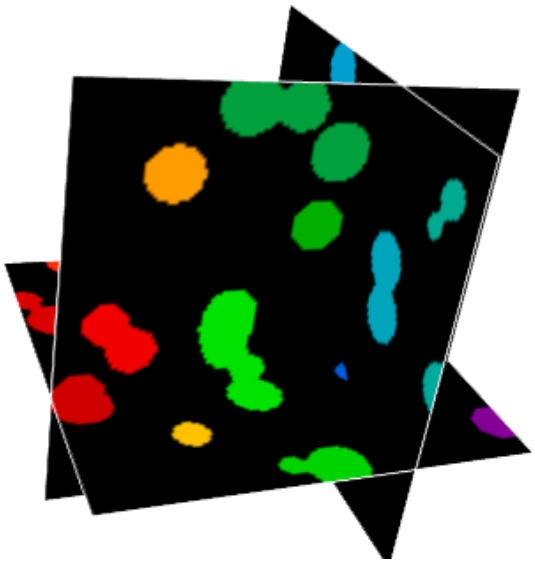


Versatile use for 2D, 2D-RGB, 3D...

```
>>> from skimage import measure
>>> labels_2d = measure.label(image_2d)
>>> labels_3d = measure.label(image_3d)
```







Try it out! `http://scikit-image.org/`

scikit-image
image processing in python

Feedback welcome

`github.com/scikit-image/scikit-image`

Please cite the paper ☺

scikit-image: image processing in Python

Stéfan van der Walt[1], Johannes L. Schönberger[2], Juan Nunez-Iglesias[3], François Boulogne[4], Joshua D. Warner[5], Neil Yager[6], Emmanuelle Gouillart[7], Tony Yu[8], the scikit-image contributors

Published June 19, 2014

PubMed 25024921

PeerJ Picks · 2015 Part of the PeerJ PeerJ Picks 2015 Collection