Image processing with `scikit-image`

Emmanuelle Gouillart
joint Unit CNRS/Saint-Gobain SVI
and the `scikit-image` team

@EGouillart

SAINT-GOBAIN

scikit-image
image processing in python

What is scikit-image?

An **open-source** (BSD)

generic **image processing library**

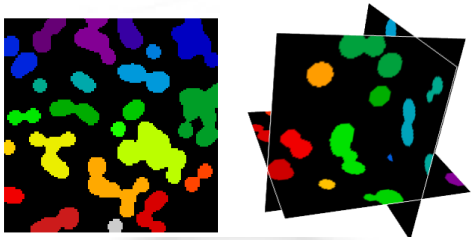
for the **Python** language
(and **NumPy** data arrays)

What is `scikit-image`?

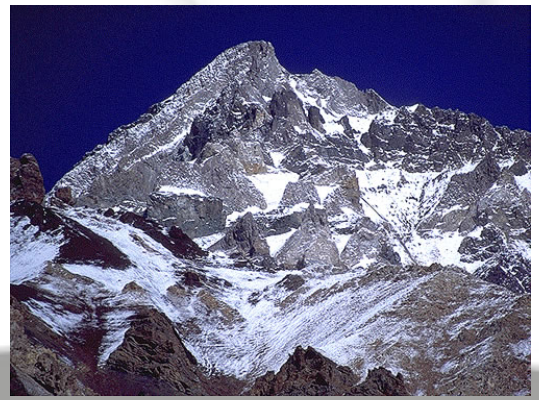
An **open-source** (BSD)

generic **image processing library**

for the **Python** language
(and **NumPy** data arrays)



for 2D & **3D** images

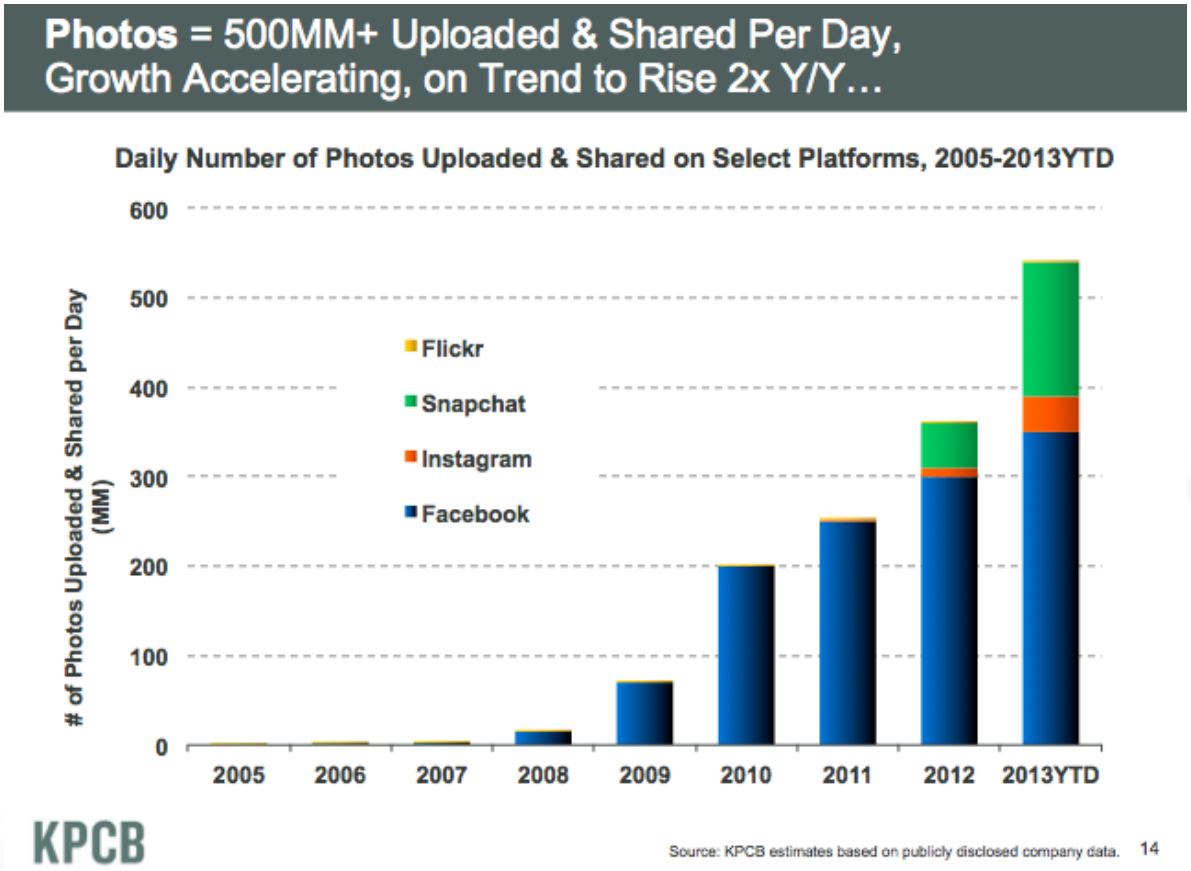


simple API & **gentle learning** curve

The vision

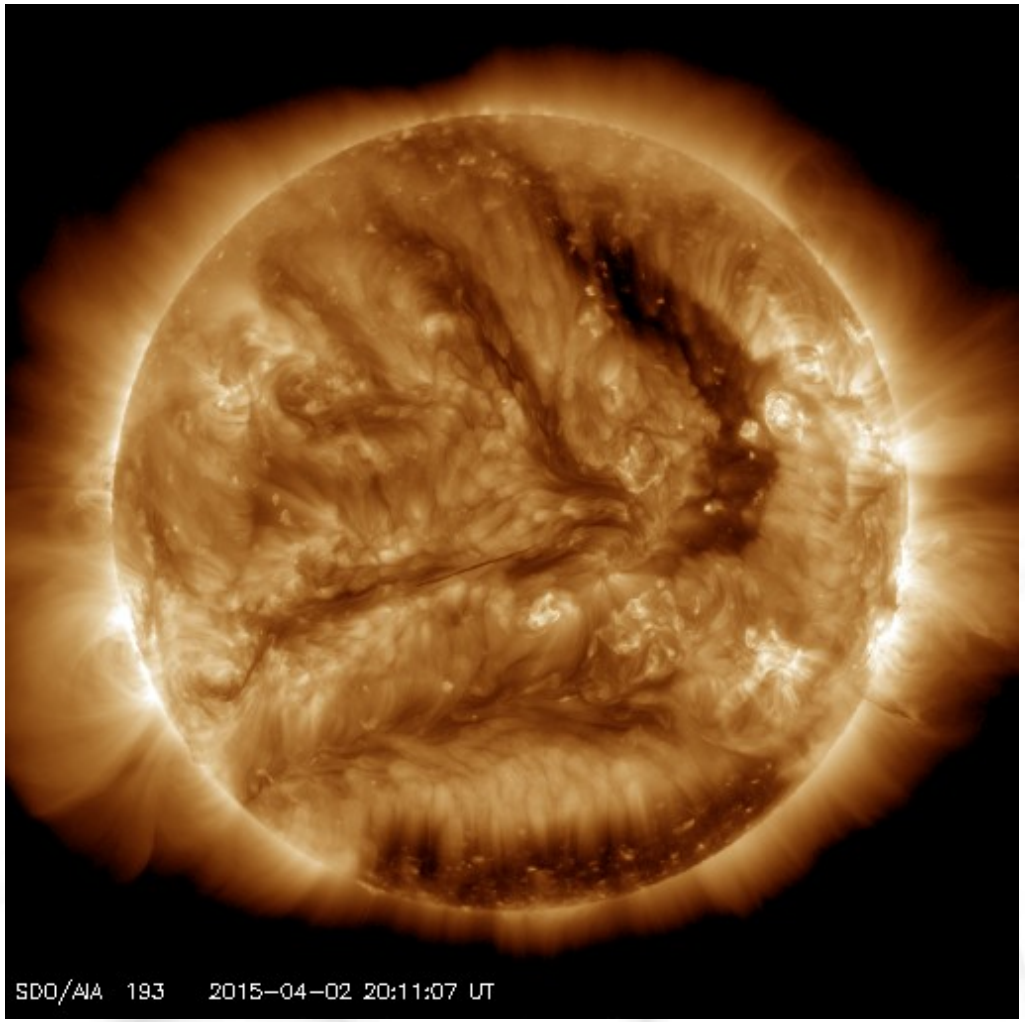
Image processing made
- easy
- instructive and didactic
- versatile and scalable

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

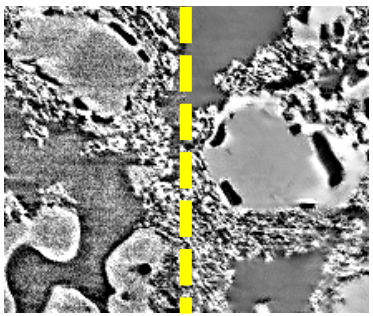
http://sdo.gsfc.nasa.gov/

What is image processing?

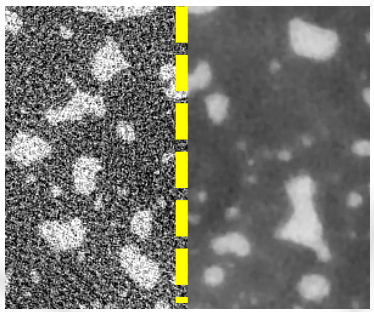
Manipulations of images to

- Transform them for other purposes: color balance, enhancement, filtering, inpainting, ...
- Extract information from them: classification, measurements...

a) filtering

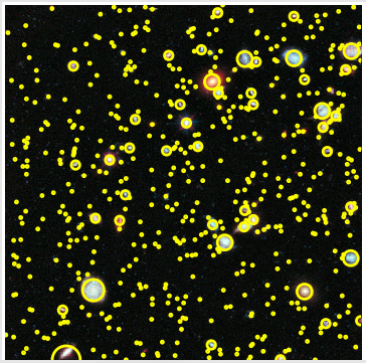


non-local means denoising

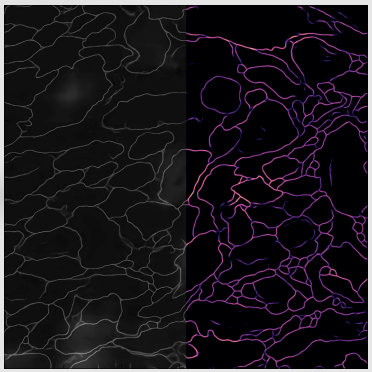


total variation denoising

b) feature extraction

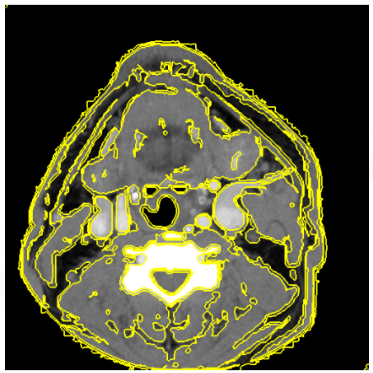


blob detection

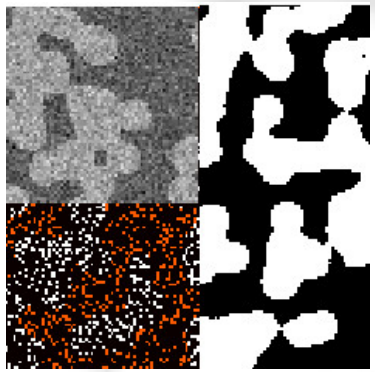


ridge detection

c) segmentation

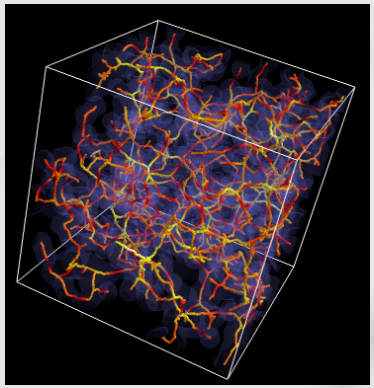


super-pixels

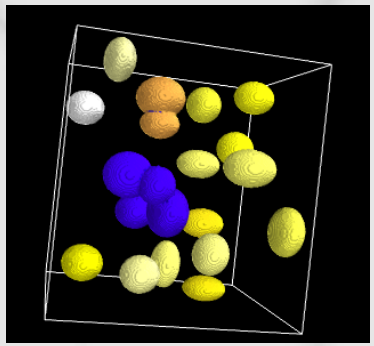


segmentation from markers

d) measures



skeleton & local diameter



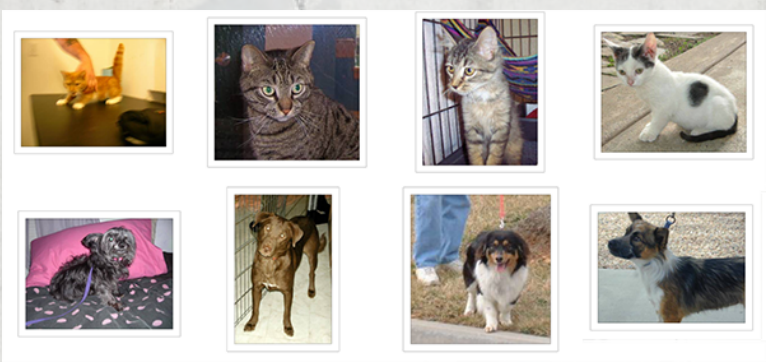
particle properties

Traditional image processing: no future?

Traditional image processing: no future?

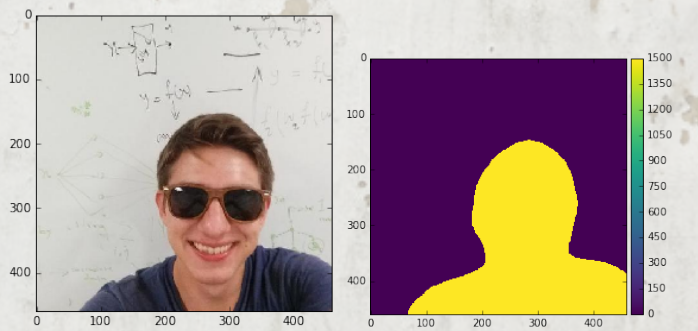
Is deep learning the universal solution?

Image classification



From Keras blog

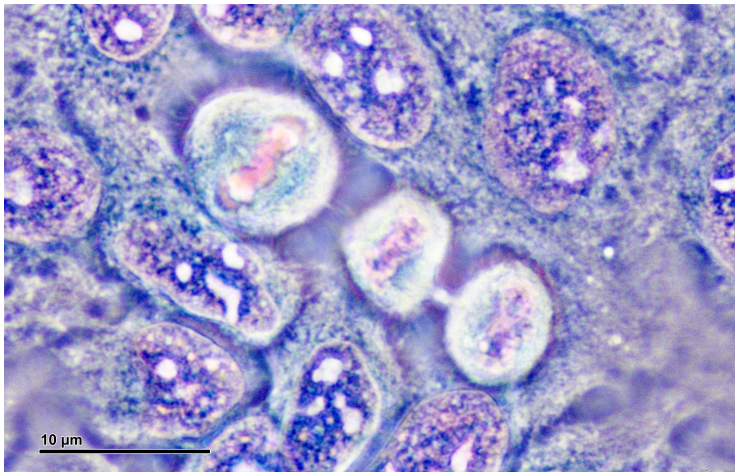
Image segmentation

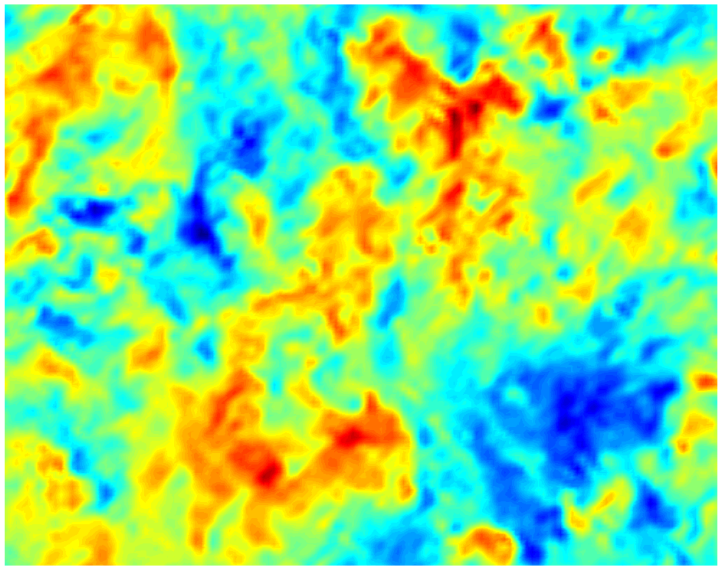


FCN, from Daniil Pakhomov's blog

Still room for other tools!

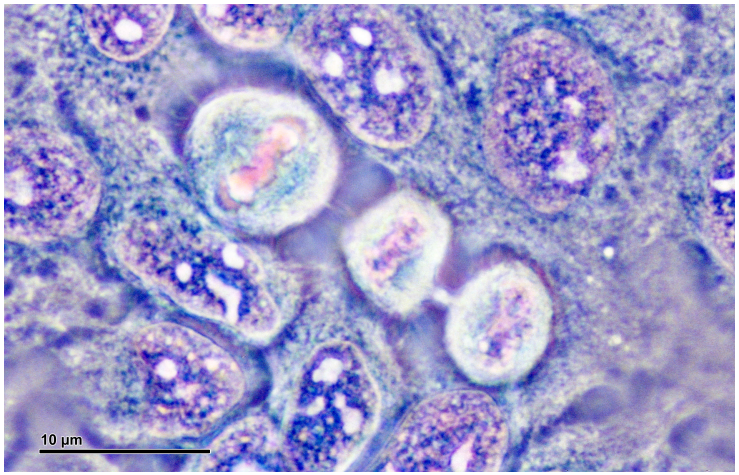
Non-natural images

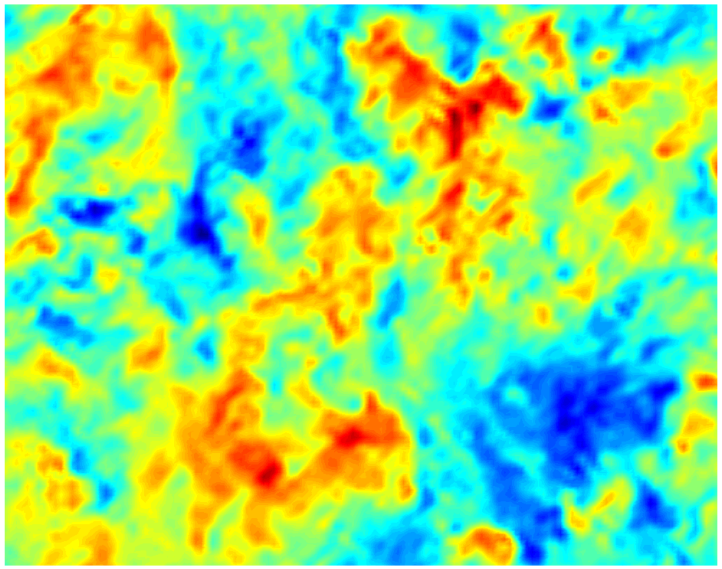




Still room for other tools!

Non-natural images





Avoid sledgehammer approach

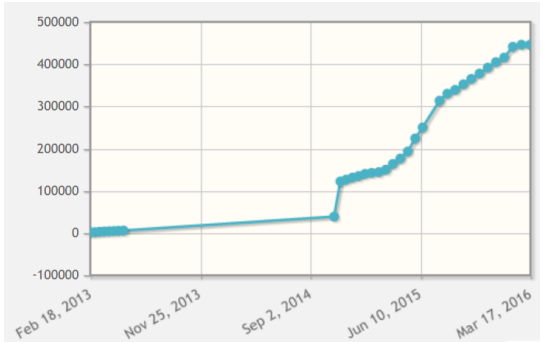
simple operations: image resizing, histogram equalization, ...

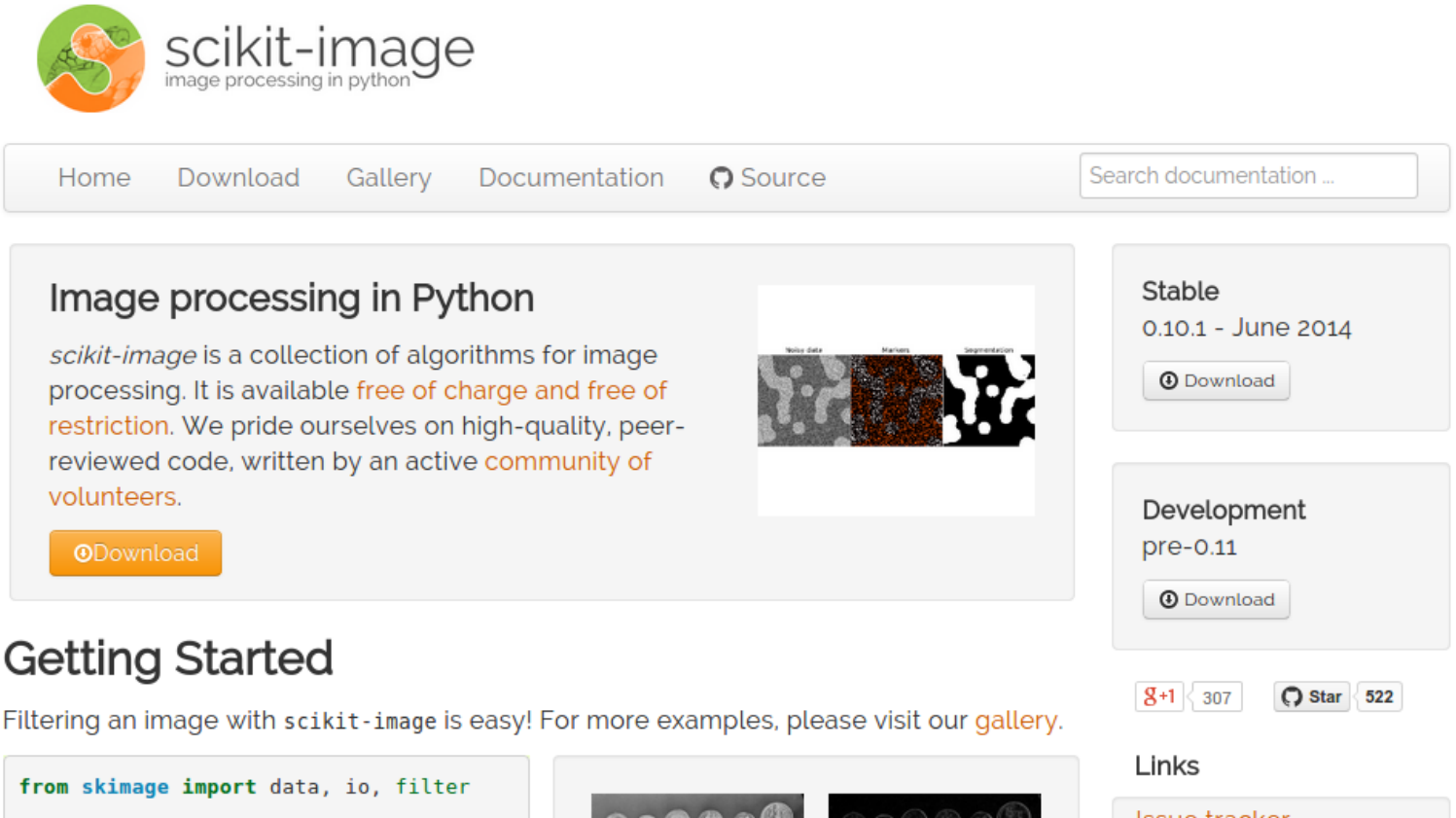
pre-processing for machine learning

Datasheet

Package statistics

- `http://scikit-image.org/`
- Release 0.13 (1 - 2 release per year)
- Among 1000 best ranked packages on PyPi
- 30000 unique visitors / month





Outline

- `scikit-image` and the `scipy` ecosystem
- Features and documentation
- People and teams
- Challenges for the future

The Scientific Python ecosystem

image file(s)

skimage.io

PyTables

Development environment

IP[y] spyder jupyter

NumPy array

1 75 ... 32
⋮
10 3 ... 2

scikit-image
image processing in python

skimage.filters
skimage.restoration
skimage.segmentation
⋮

machine learning

scikit learn

8 23 ... 36
⋮
1 13 ... 0

12 43 ... 6
⋮
66 ... 97 1

1 41 ... 98
⋮
38 75 ... 6

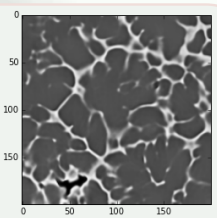
8 5 ... 73
⋮
78 98 ... 9

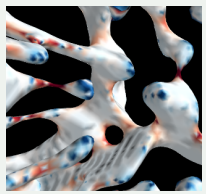
4 10 ... 55
⋮
10 3 ... 9

skimage.measure
skimage.feature

$[39 \ldots 5]$

matplotlib

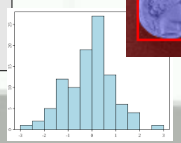


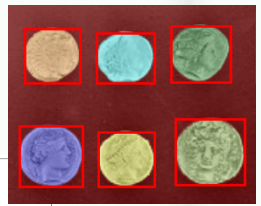


Mayavi

visualization

image properties
object properties

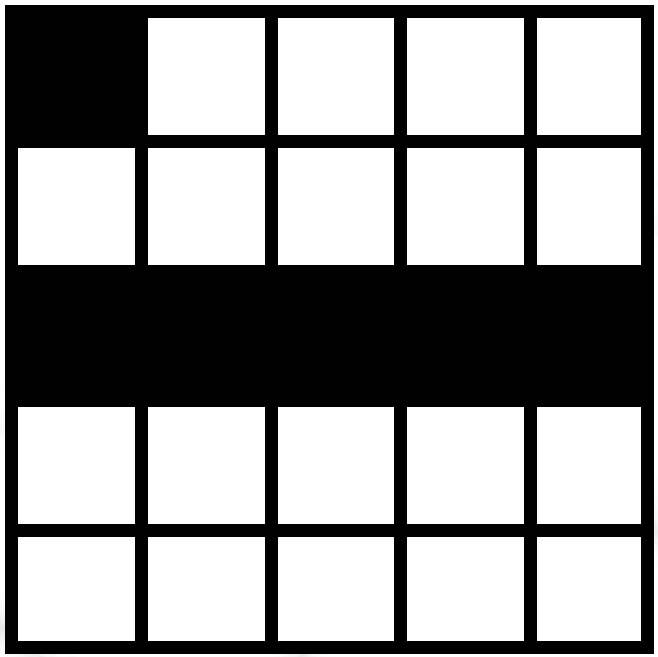




Manipulating images as numerical (numpy) arrays

- Pixels are arrays elements

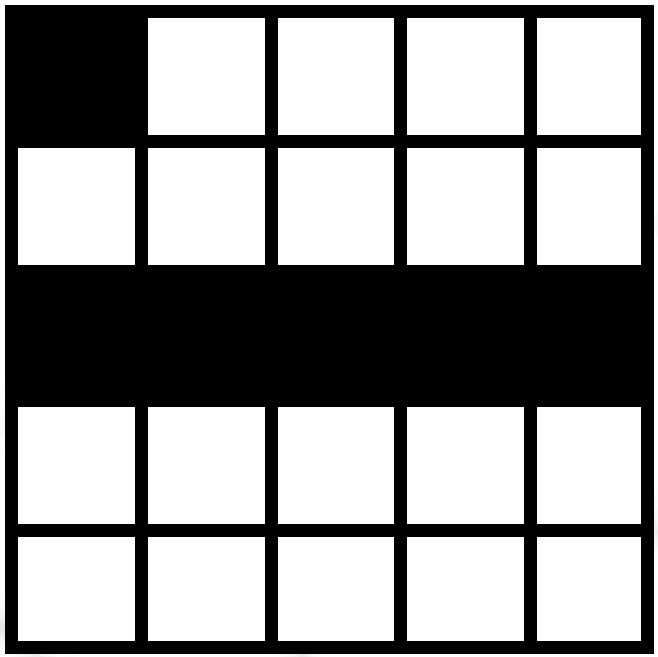
```
import numpy as np
image = np.ones((5, 5))
image[0, 0] = 0
image[2, :] = 0
```



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))
```

NumpPy: Python objects for numerical arrays

Multi-dimensional numerical data container (based on compiled code)

\+ utility functions to create/manipulate them

```
>>> a = np.random.random_integers(0, 1, (2, 2, 2))
>>> a
array([[[0, 1],
        [1, 0]],

       [[0, 0],
        [0, 1]]])
>>> a.shape, a.dtype
((2, 2, 2), dtype('int64'))
```

NumpPy: Python objects for numerical arrays

Multi-dimensional numerical data container (based on compiled code)
+ utility functions to create/manipulate them

```
>>> a = np.random.random_integers(0, 1, (2, 2, 2))
>>> a
array([[[0, 1],
        [1, 0]],

       [[0, 0],
        [0, 1]]])
>>> a.shape, a.dtype
((2, 2, 2), dtype('int64'))
```

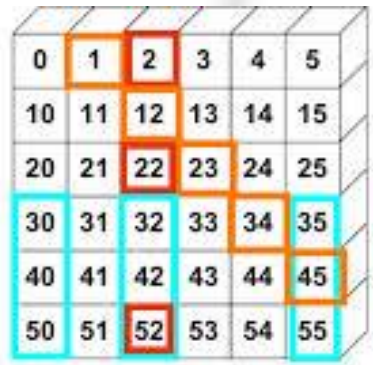
Efficient and versatile data access

indexing and slicing

```
>>> a[0,3:5]
array([3,4])

>>>-a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,22,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

fancy indexing

```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35]
       [40, 42, 45]])
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```



NumPy Cheatsheet

array initialization

```
np.array([2, 3, 4])            # direct initial
np.empty(20, dtype=np.float32) # single precisio
np.zeros(200)                  # initialize 200
np.ones((3,3), dtype=np.int32) # 3 x 3 integer
np.eye(200)                    # ones on the dia
np.zeros_like(a)               # returns array
np.linspace(0., 10., 100)      # 100 points fro
np.arange(0, 100, 2)           # points from 0
np.logspace(-5, 2, 100)        # 100 log-spaced
np.copy(a)                     # copy array to
```

reading/ writing files

```
np.fromfile(fname/object, dtype=np.float32, count
np.loadtxt(fname/object, skiprows=2, delimiter=',
```

indexing

```
a = np.arange(100)          # initialization with
a[:3] = 0                   # set the first three
a[1:5] = 1                  # set indices 1-4 to
a[start:stop:step]          # general form of ind
a[None, :]                  # transform to column
a[[1, 1, 3, 8]]             # return array with
a = a.reshape(10, 10)       # transform to 10 x 1
a.T                         # return transposed
np.transpose(a, (2, 1, 0))  # transpose array to
a[a < 2]                    # returns array that
```

boolean arrays

```
a < 2                       # returns array wit
(a < 2) & (b > 10)          # elementwise logic
(a < 2) | (b > 10)          # elementwise logic
~a                          # invert boolean a
```

array properties and operations

```
a.shape                 # a tuple with the length
len(a)                  # length of axis 0
a.ndim                  # number of dimensions (a
a.sort(axis=1)          # sort array along axis
a.flatten()             # collapse array to one d
a.conj()                # return complex conjugate
a.astype(np.int16)      # cast to integer
np.argmax(a, axis=2)    # return index of maximum
np.cumsum(a)            # return cumulative sum
np.any(a)               # True if any element is
np.all(a)               # True if all elements are
np.argsort(a, axis=1)   # return sorted index arra
```

elementwise operations and math functions

```
a * 5               # multiplication with scalar
a + 5               # addition with scalar
a + b               # addition with array b
a / b               # division with b (np.NaN for
np.exp(a)           # exponential (complex and rea
np.power(a,b)       # a to the power b
np.sin(a)           # sine
np.cos(a)           # cosine
np.arctan2(y,x)     # arctan(y/x)
np.arcsin(x)        # arcsin
np.radians(a)       # degrees to radians
```

inner / outer products

```
np.dot(a, b)                    # inner matr
np.einsum('ijkl,klmn->ijmn', a, b)  # einstein su
np.sum(a, axis=1)               # sum over a
np.abs(a)                       # return arra
a[None, :] + b[:, None]         # outer sum
a[None, :] * b[:, None]         # outer produ
np.outer(a, b)                  # outer produ
np.sum(a * a.T)                 # matrix norm
```

random variables

```
np.random.normal(loc=0, scale=2, size=100)  # 10
np.random.seed(23032)                       # re
np.random.rand(200)                         # 20
np.random.uniform(1, 30, 200)               # 20
np.random.random_integers(1, 15, 300)       # 30
```

rounding

```
np.ceil(a)   # rounds to nearest upper int
np.floor(a)  # rounds to nearest lower int
np.round(a)  # rounds to neares int
```

Adapted from the Scientific python cheat sheet of IPGP
https://github.com/IPGP/scientific_python_cheat_sheet.

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(camera_array,
    sigma=5)
>>> type(filtered_array)
<type 'numpy.ndarray'>
>>> import matplotlib.pyplot as plt
>>> plt.imshow(filtered_array, cmap='gray')
```

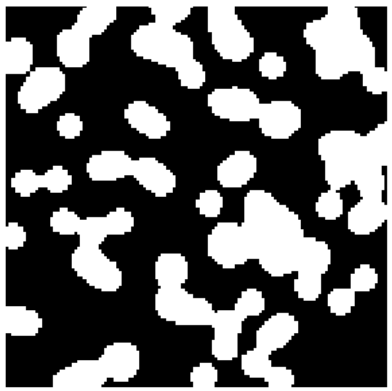
How we simplified the API

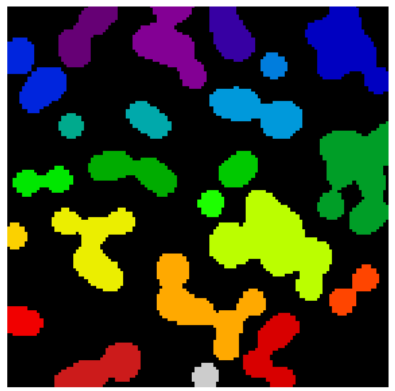
- Before 2013

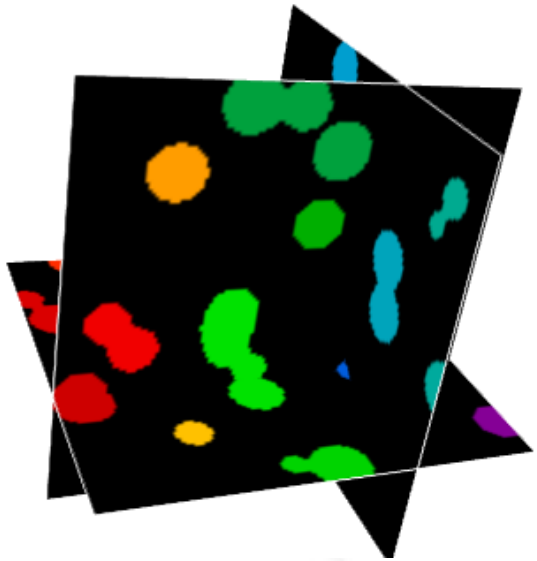
```
>>> from skimage import io, filters
>>> camera_array = io.imread('camera_image.png')
>>> type(camera_array)
Image...
>> camera.max()
Image(255, dtype=uint8)
```

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

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







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

```
def quickshift(image, ratio=1.0, kernel_size=5,
    max_dist=10,
                   sigma=0, random_seed=42):
    """Segments image using quickshift clustering in
        Color-(x,y) space.

    ...
    """

    image = img_as_float(np.atleast_3d(image))
    ...
```

An API relying mostly on functions

```
feature.canny(image, sigma=1.0, low_threshold=None,
    high_threshold=None, mask=None, use_quantiles=
    False)

Edge filter an image using the Canny algorithm.

Parameters
----------
image : 2D array
    Greyscale input image to detect edges on; can be
        of any dtype.
sigma : float
    Standard deviation of the Gaussian filter.
low_threshold : float
    Lower bound for hysteresis thresholding (linking
        edges).
    If None, low_threshold is set to 10% of dtype's
        max.
high_threshold : float
    Upper bound for hysteresis thresholding (linking
        edges).
    If None, high_threshold is set to 20% of dtype's
```

One filter = one function

Use keyword argument for parameter tuning

`MCP`

---

class skimage.graph. **MCP** (*costs*, *offsets=None*, *fully_connected=True*, *sampling=None*)

Bases: `object`

A class for finding the minimum cost path through a given n-d costs array.

Given an n-d costs array, this class can be used to find the minimum-cost path through that array from any set of points to any other set of points. Basic usage is to initialize the class and call find_costs() with a one or more starting indices (and an optional list of end indices). After that, call traceback() one or more times to find the path from any given end-position to the closest starting index. New paths through the same costs array can be found by calling find_costs() repeatedly.

The cost of a path is calculated simply as the sum of the values of the costs array at each point on the path. The class MCP_Geometric, on the other hand, accounts for the fact that diagonal vs. axial moves are of different lengths, and weights the path cost accordingly.

Array elements with infinite or negative costs will simply be ignored, as will paths whose cumulative cost overflows to infinite.

Parameters:

costs : ndarray

offsets : iterable, optional

> A list of offset tuples: each offset specifies a valid move from a given n-d position. If not provided, offsets corresponding to a singly- or fully-connected n-d neighborhood will be constructed with make_offsets(), using the fully_connected parameter value.

API of scikit-learn

3.6.1.1. Learning and Predicting

Now that we've got some data, we would like to learn from it and predict on new one. In `scikit-learn`, we learn from existing data by creating an `estimator` and calling its `fit(X, Y)` method.

```
>>> from sklearn import svm
>>> clf = svm.LinearSVC()
>>> clf.fit(iris.data, iris.target) # learn from the data
LinearSVC(...)
```

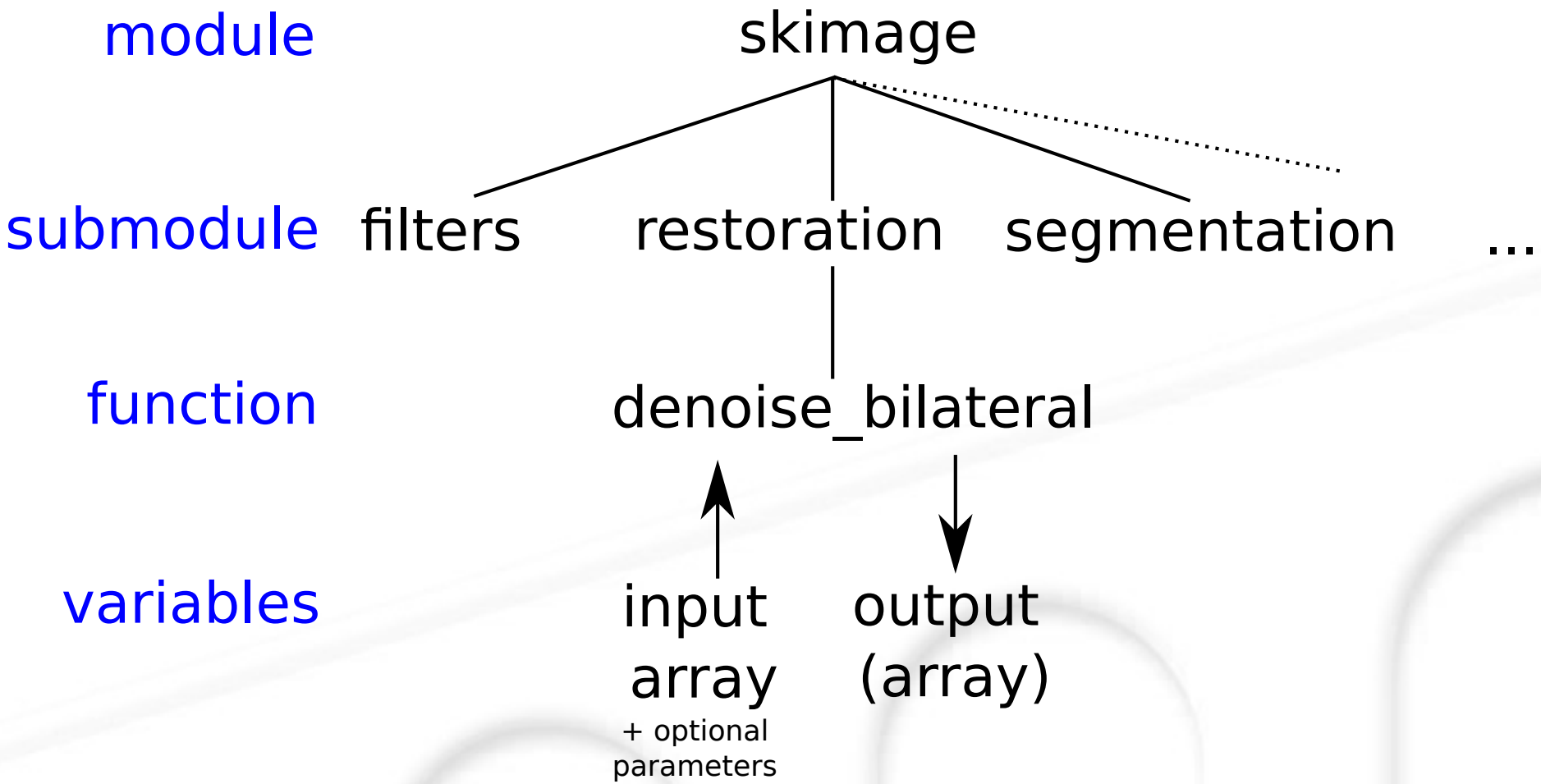
Once we have learned from the data, we can use our model to predict the most likely outcome on unseen data:

```
>>> clf.predict([[ 5.0,  3.6,  1.3,  0.25]])
array([0])
```

Note: We can access the parameters of the model via its attributes ending with an underscore:

```
>>> clf.coef_
array([[ 0...]])
```

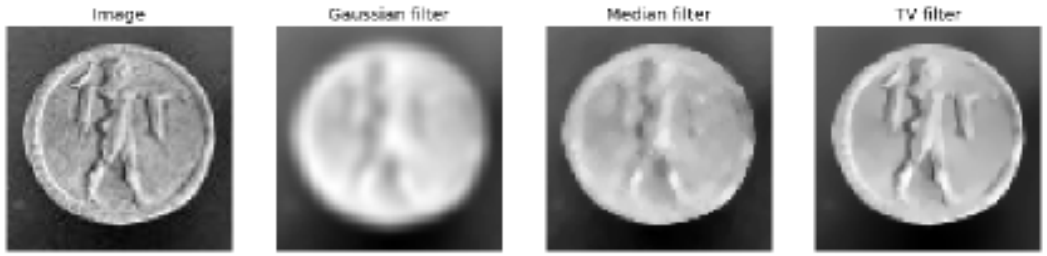
API of scikit-image

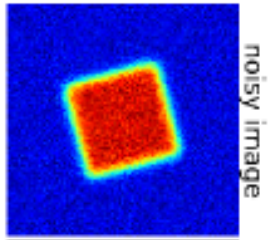


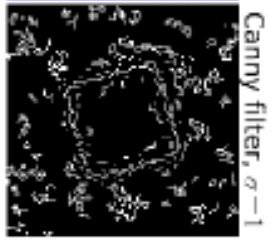
Converging to a coherent API

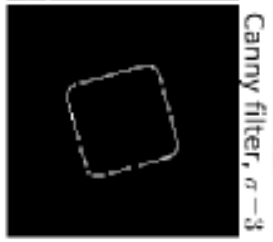
- Module: `restoration`
  - References
  - denoise_bilateral
  - denoise_nl_means
  - denoise_tv_bregman
  - denoise_tv_chambolle
  - nl_means_denoising
  - richardson_lucy
  - unsupervised_wiener
  - unwrap_phase
  - wiener

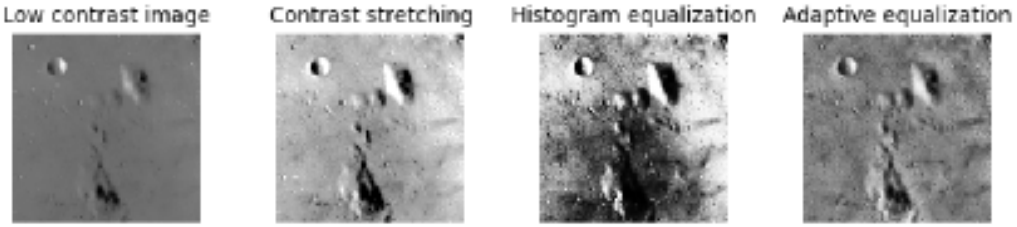
Filtering: transforming image data

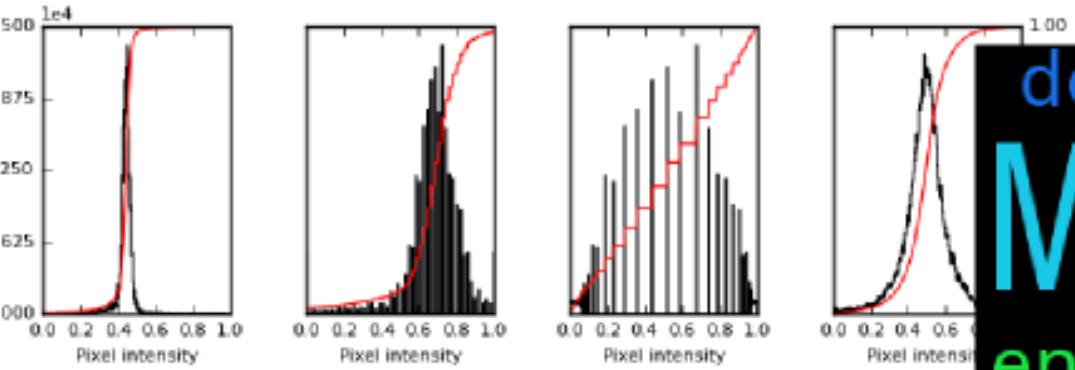


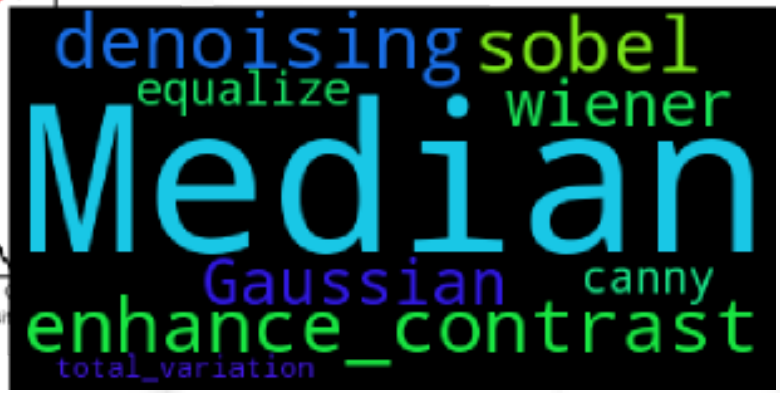






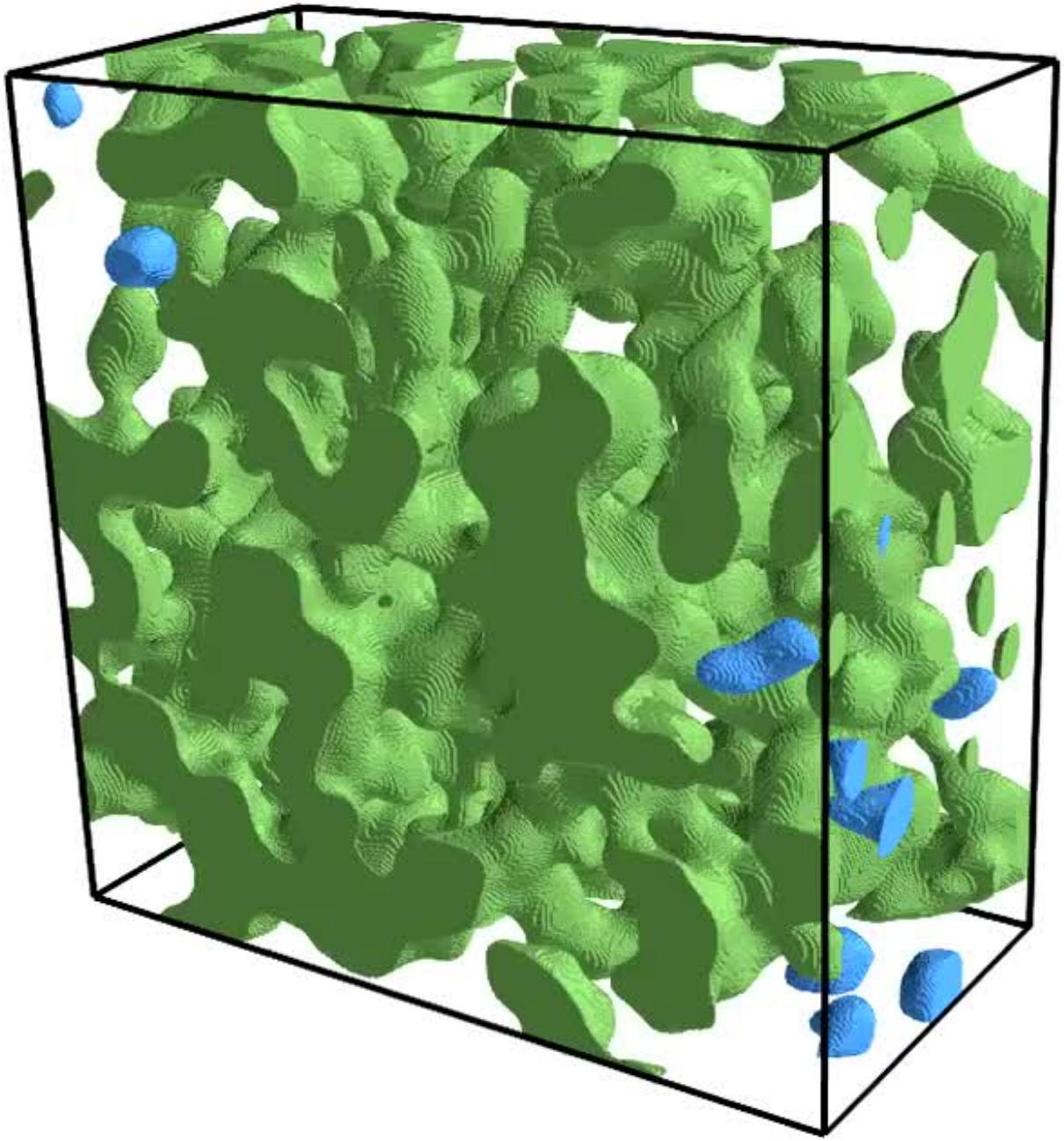


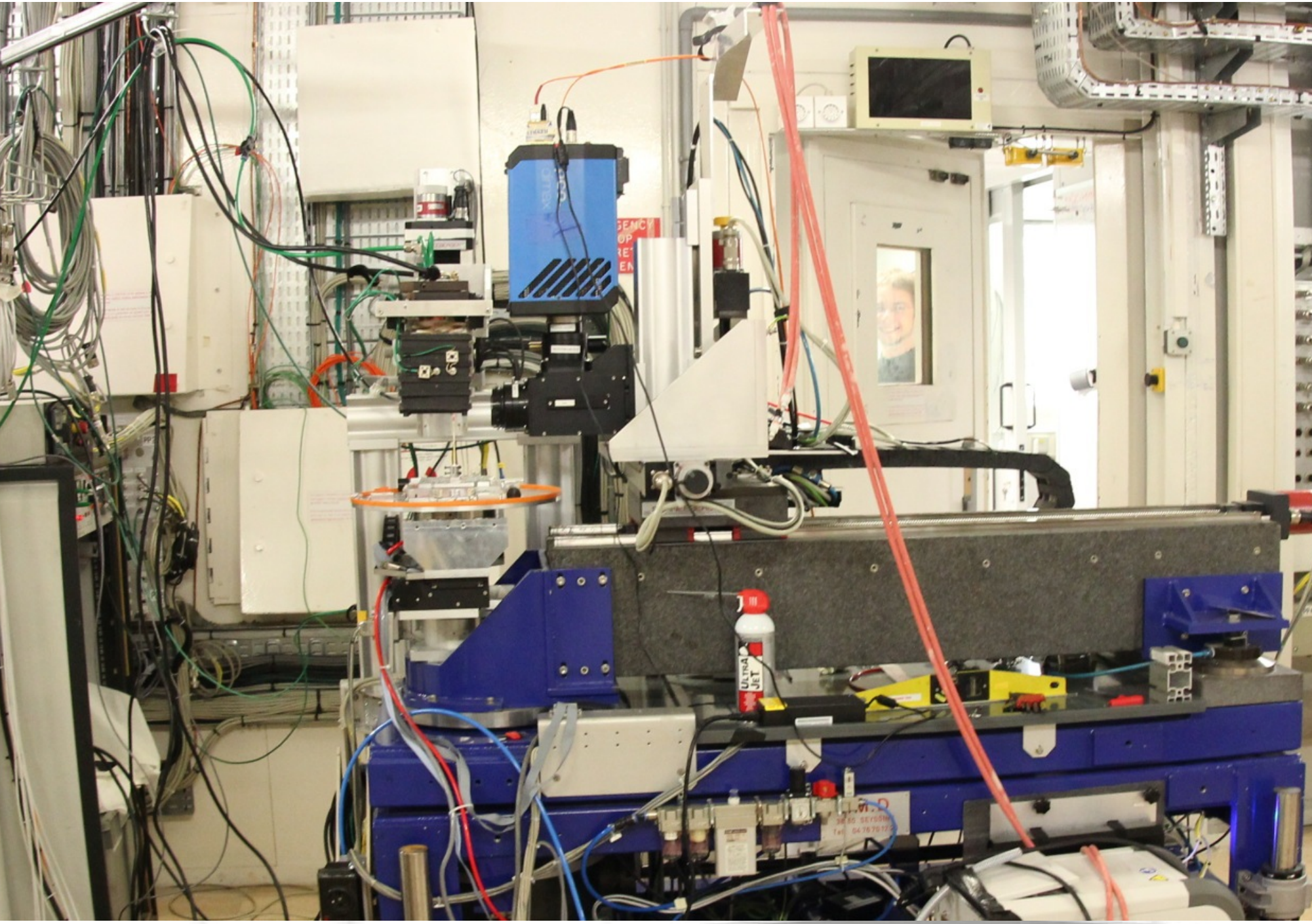




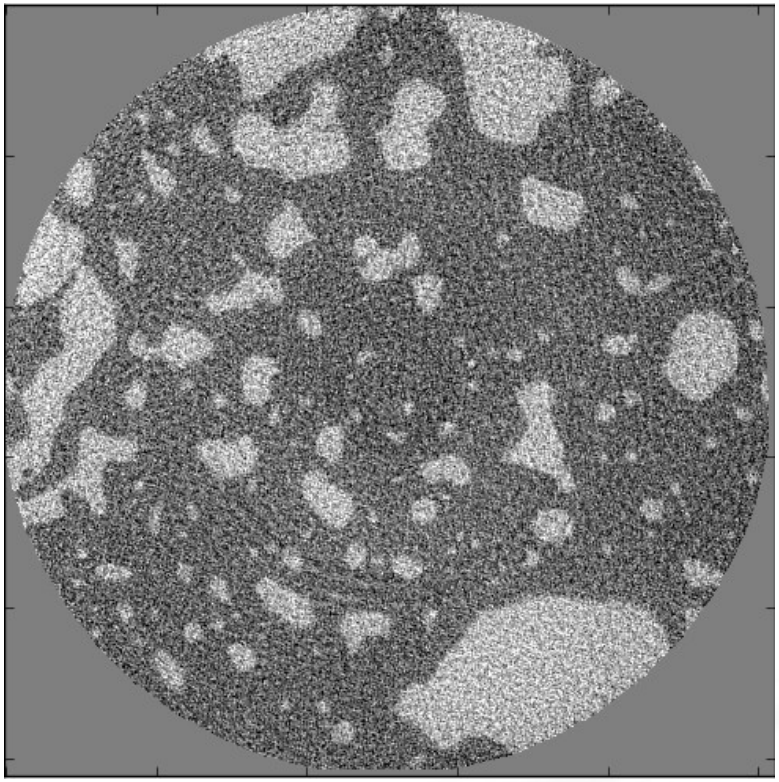
`skimage.filters`, `skimage.exposure`, `skimage.restoration`

In situ study of phase separation

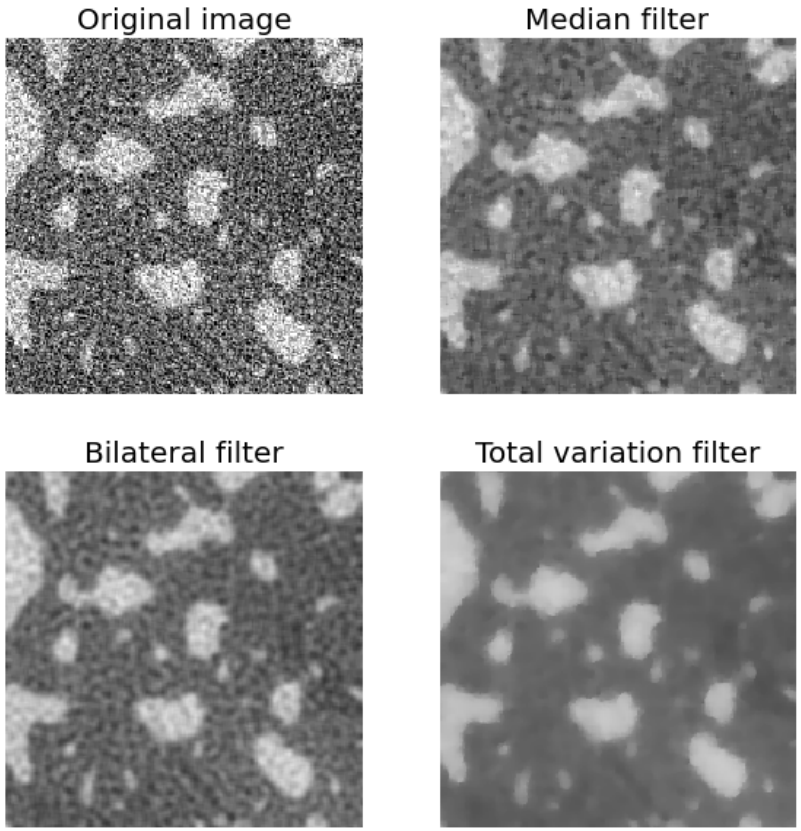



ULTRA
JET

Denoising tomography images



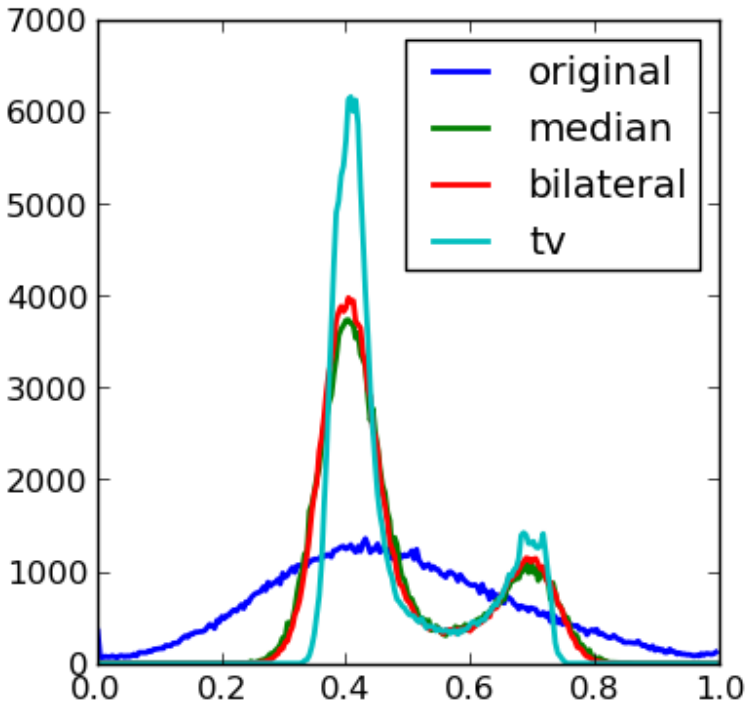
In-situ imaging of phase separation in silicate melts

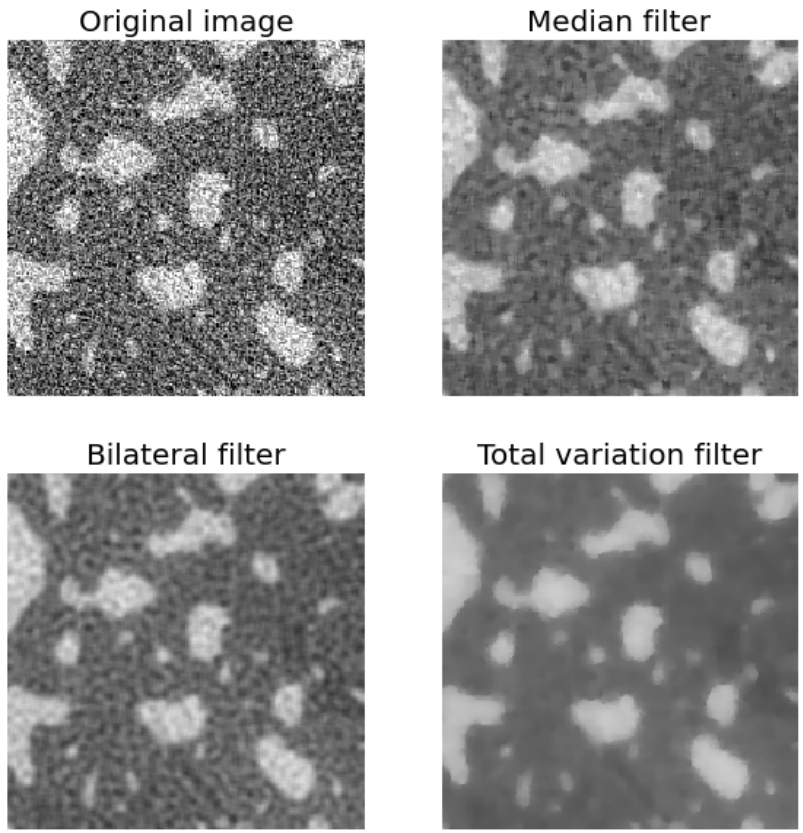


From basic (generic) to advanced (specific) filters

Denoising tomography images

Histogram of pixel values

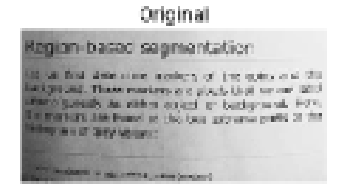


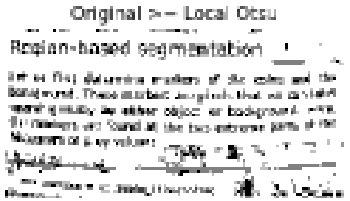


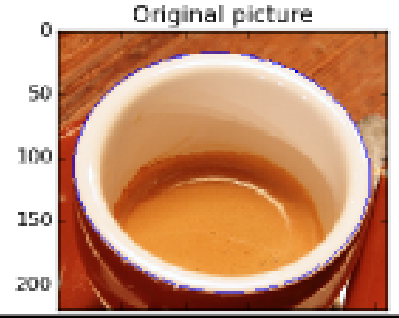
From basic (generic) to advanced (specific) filters

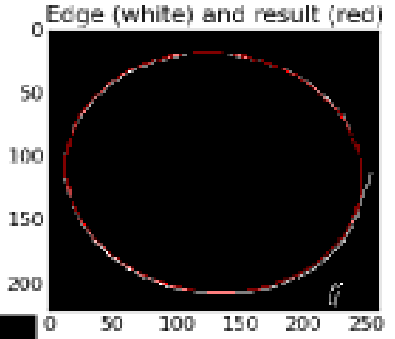
```
bilateral = restoration.denoise_bilateral(dat)
bilateral = restoration.denoise_bilateral(dat, sigma_range=2.5,
sigma_spatial=2)
tv = restoration.denoise_tv_chambolle(dat, weight=0.5)
```

Segmentation: labelling regions

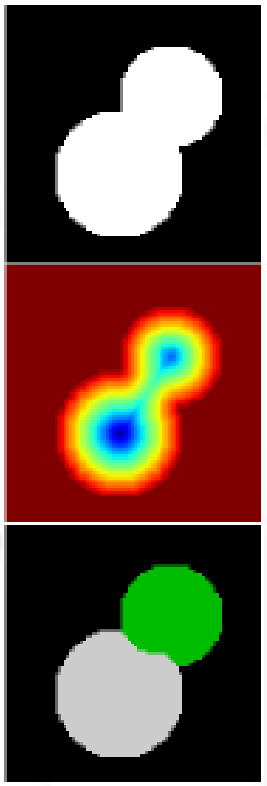




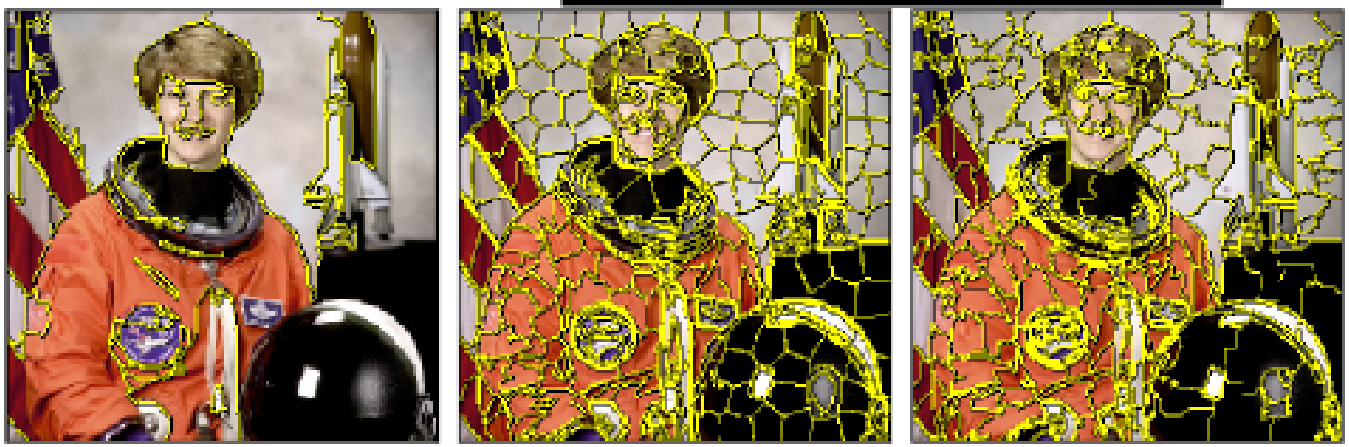




watershed
superpixel
thresholding
otsu
randomwalker



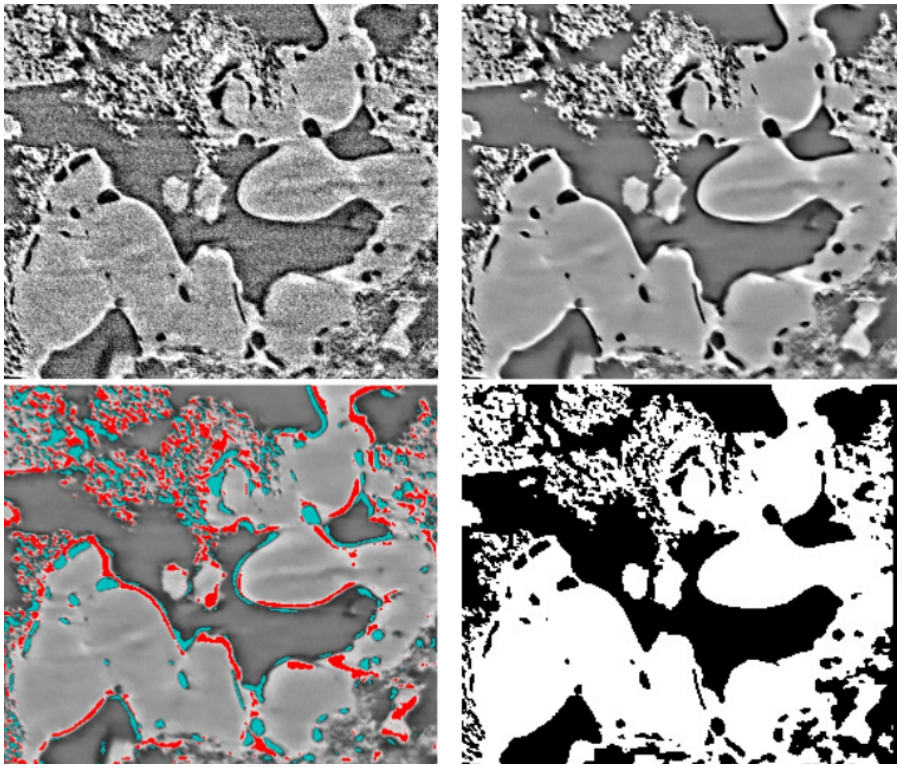
Felzenszwalbs's method



skimage.segmentation

Example: segmentation of low-constrast regions

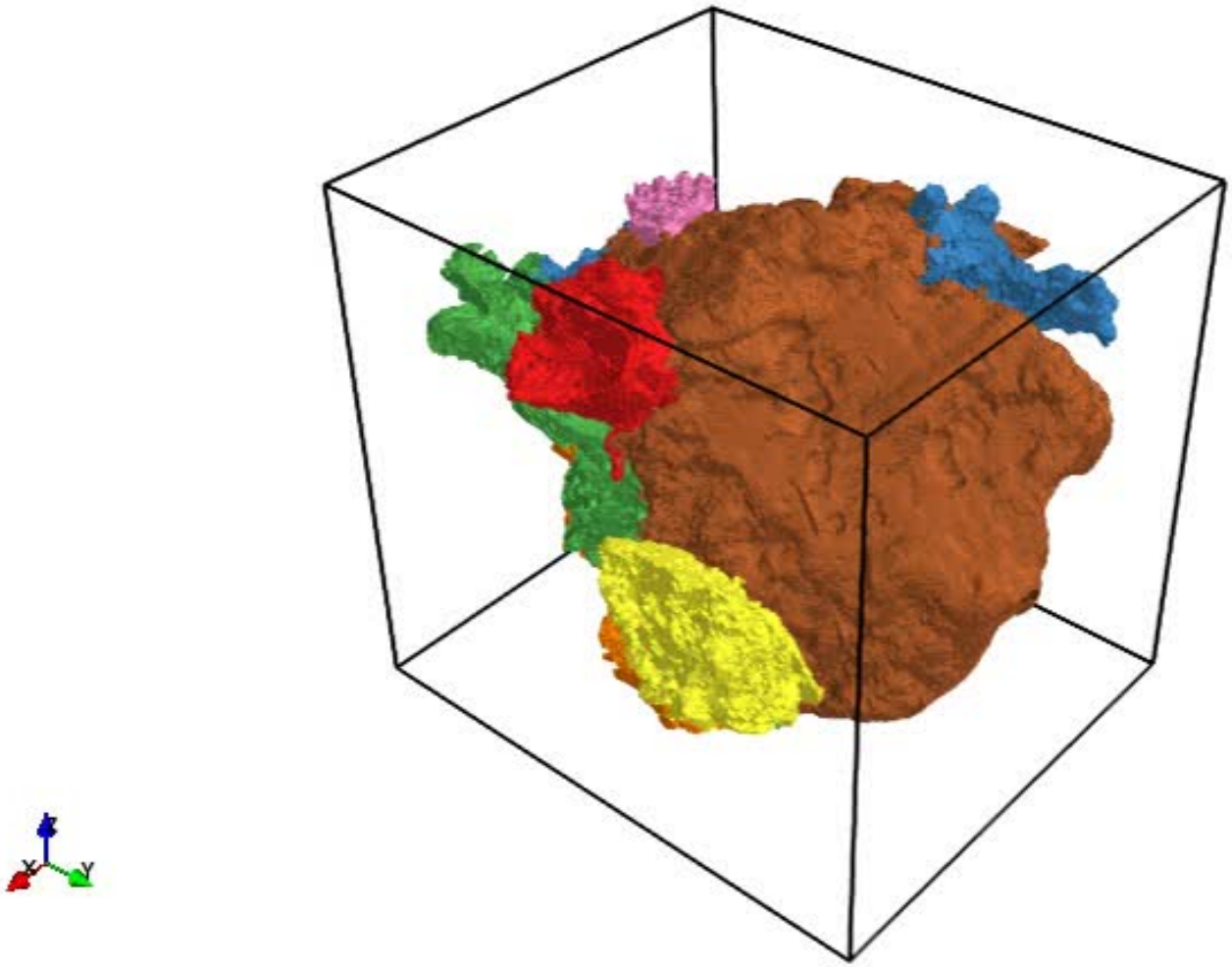
In-situ imaging of glass batch reactive melting



- Non-local means denoising to preserve texture
- Histogram-based markers extraction
- Random walker segmentation

Non-local means: average similar patches
Random walker:anisotropic diffusion from markers
Random walker less sensitive to noise than watershed, but slower

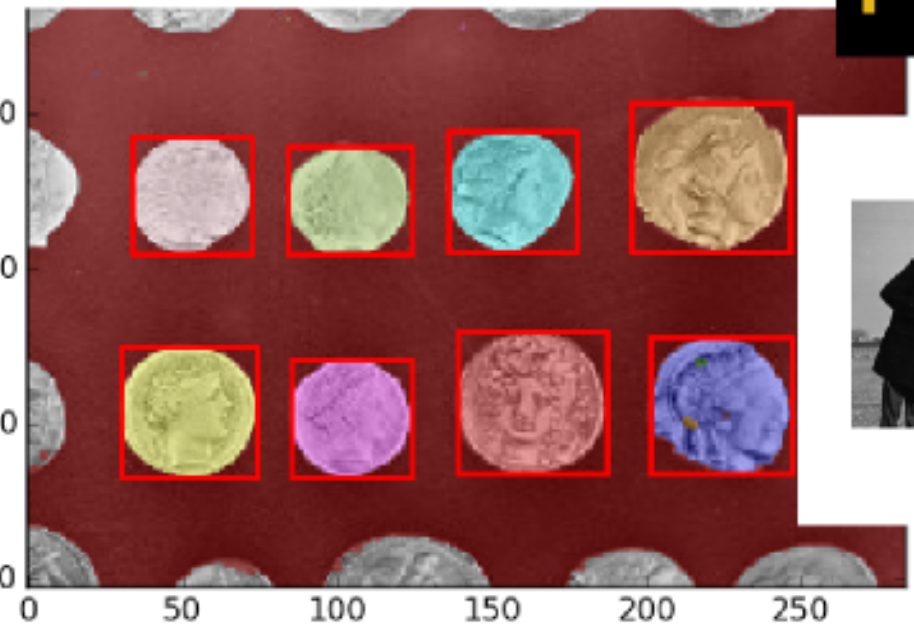
Visualizing the geometry of reactions

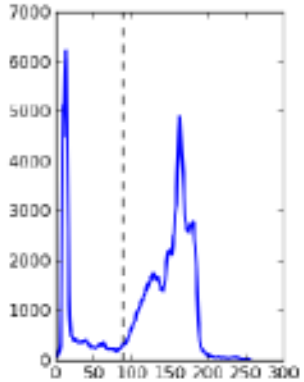


Quantifying the reacted parts of the grain

Measures on images

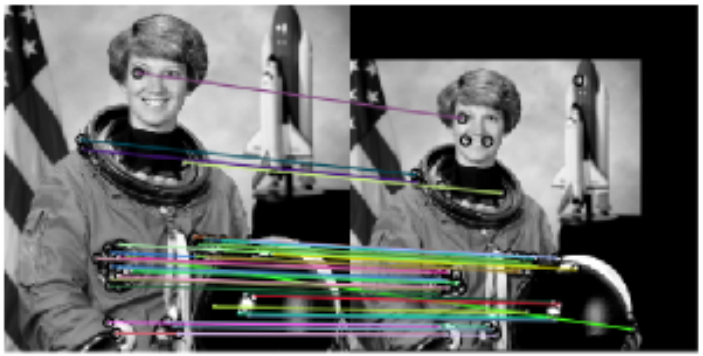
size label
measure
histogram
regionprops

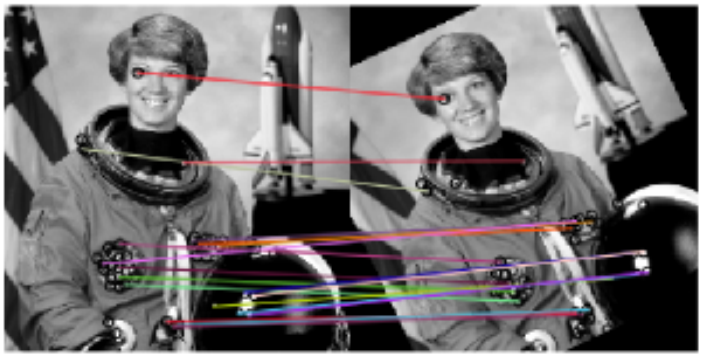


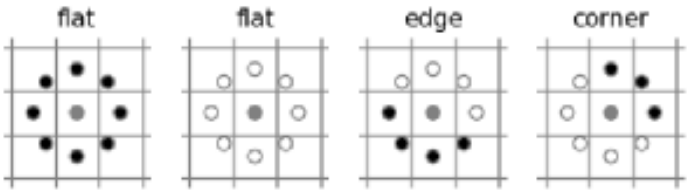


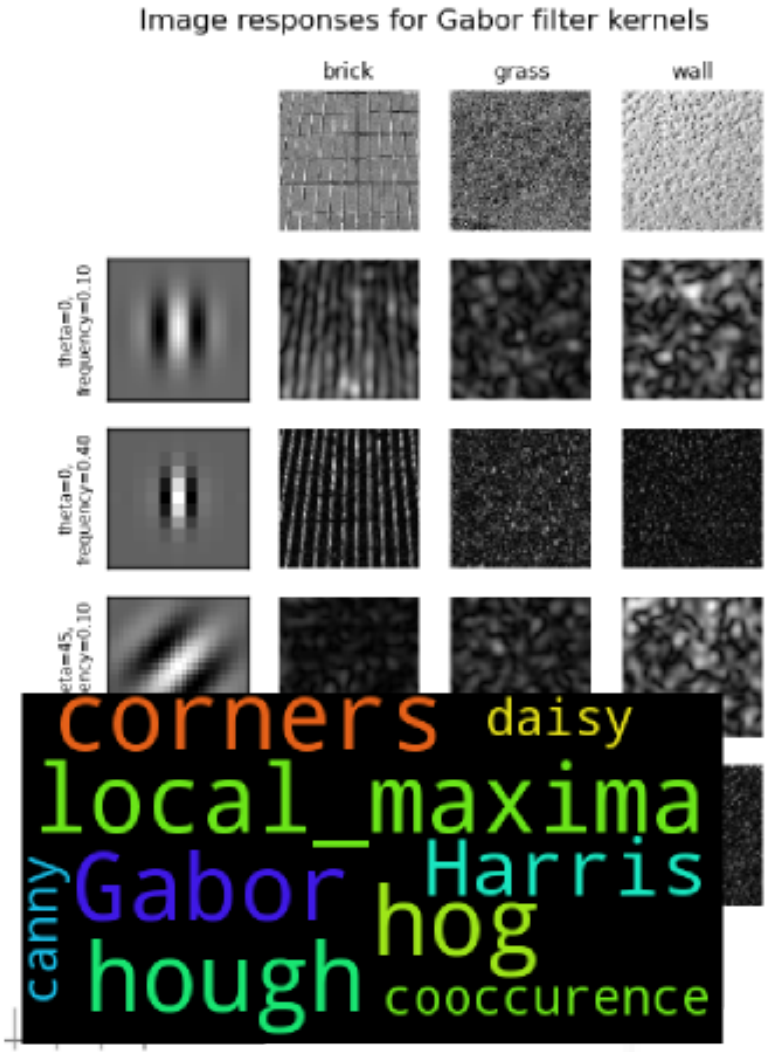
skimage.measure

Extracting features









`skimage.feature`, `skimage.filters`

Feature extraction followed by classification

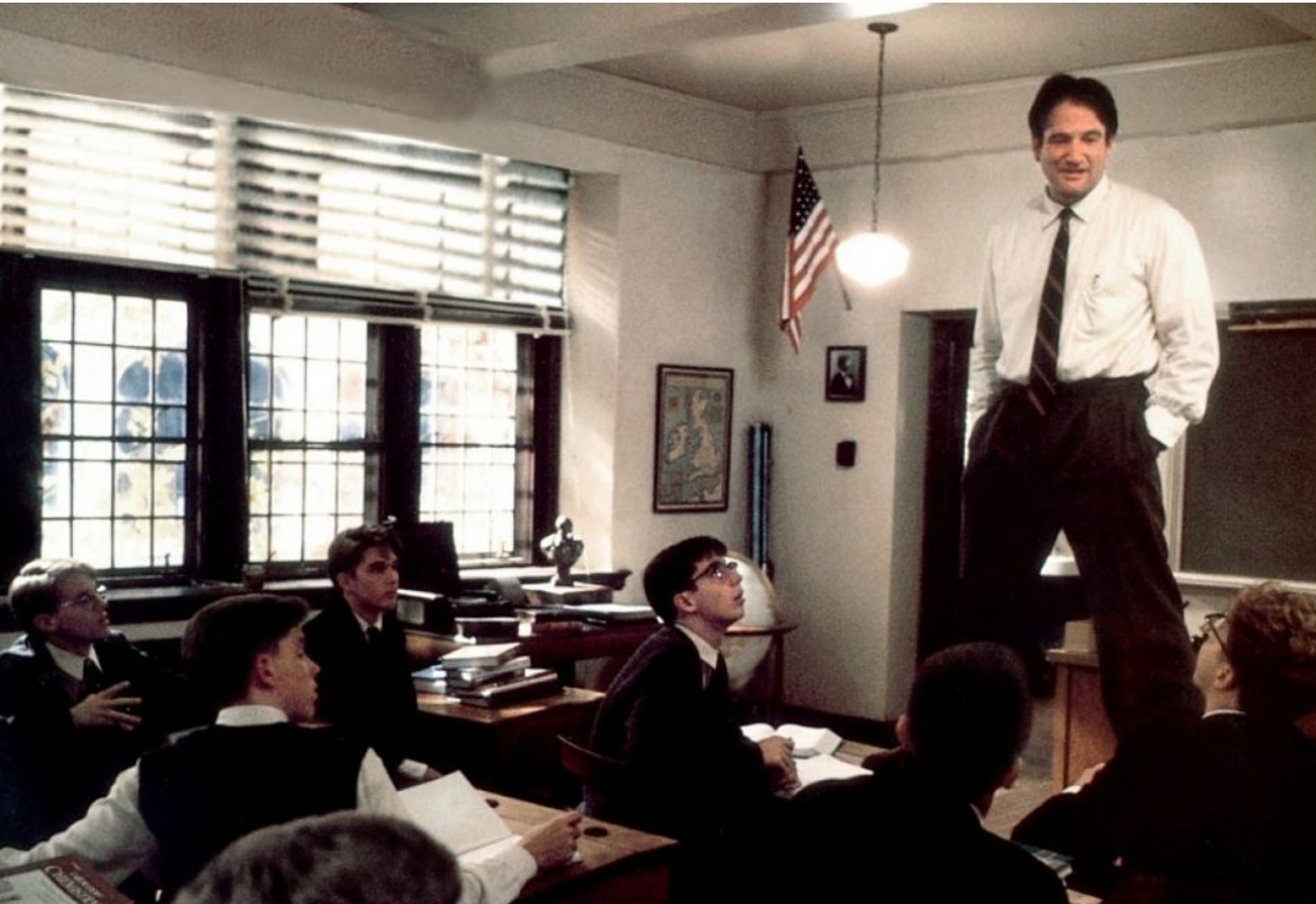
Combining `scikit-image` and `scikit-learn`

- Extract features (`skimage.feature`)
  - Gabor filters for each color channels
  - ...
- Train classifier with known regions
  - here, random forest classifier
- Classify pixels

Scipy 2017 tutorial on image processing and machine learning

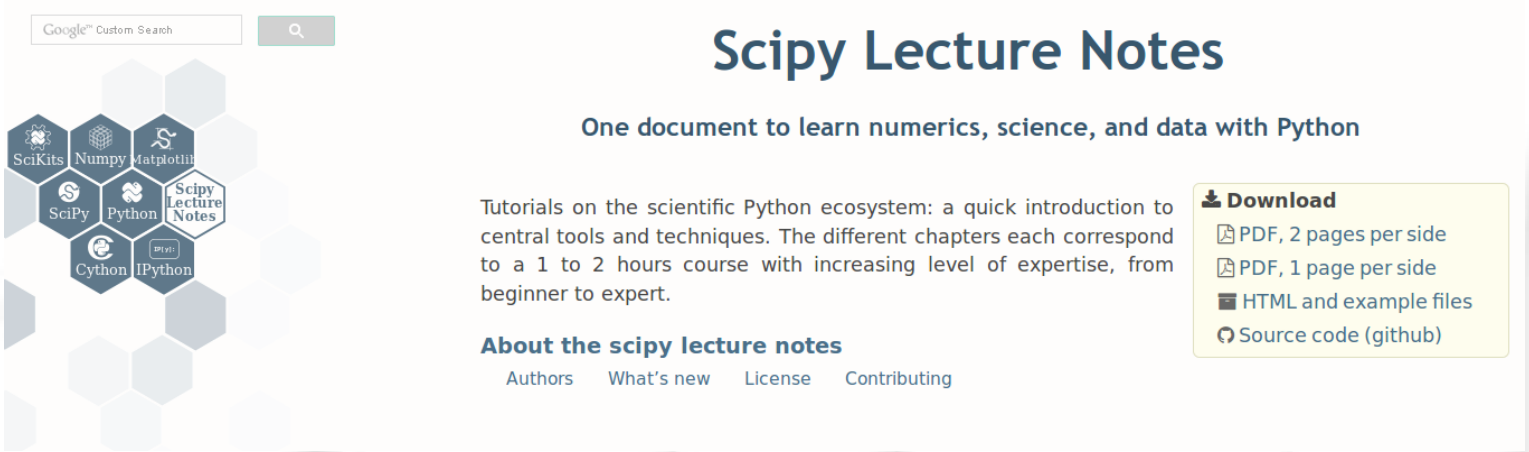
`https://github.com/scikit-image/skimage-tutorials/`

Documentation and teaching



Trainings, tutorials and narrative documentation

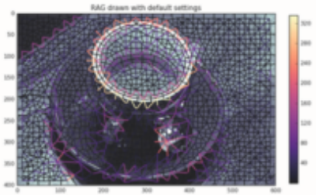
- **Trainings** at Scipy/Pydata conferences + YouTube videos
- **Scipy lecture notes** chapter on `scikit-image`
- **Collection of tutorials**: `https://github.com/scikit-image/skimage-tutorials`
- **User guide**: still work in progress



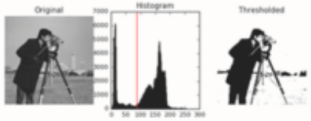
What is good documentation?

"Documenting code is like writing "Tasty!" on the side of a coffee cup. If the code isn't readable on a grey Monday morning before coffee, chuck it out and start again. What you document are APIs (...). That is fine. Explaining what this funky loop does is not fine." Pieter Hintjens

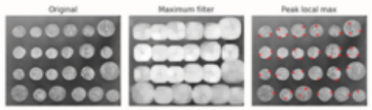
Gallery of examples



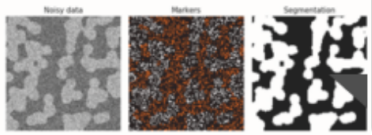
Drawing Region Adjacency Graphs (RAGs)



Thresholding

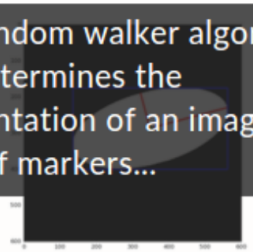


Finding local maxima

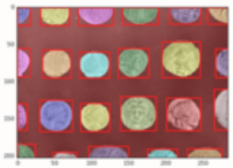


Random walker segmentation

The random walker algorithm [1]_ determines the segmentation of an image from a set of markers...

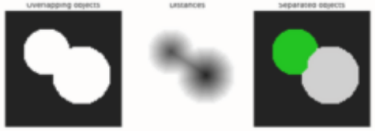


Measure region properties



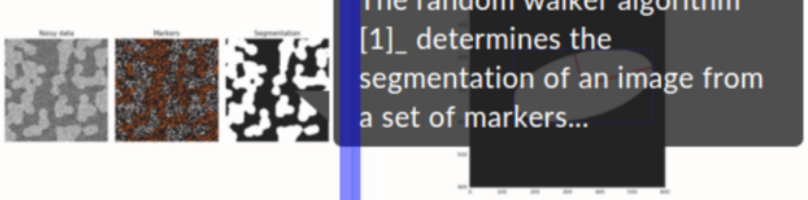
Label image regions

Comparison of segmentation and superpixel algorithms



Watershed segmentation

Markers for watershed transform



Random walker segmentation

Measure region properties

Label image regions

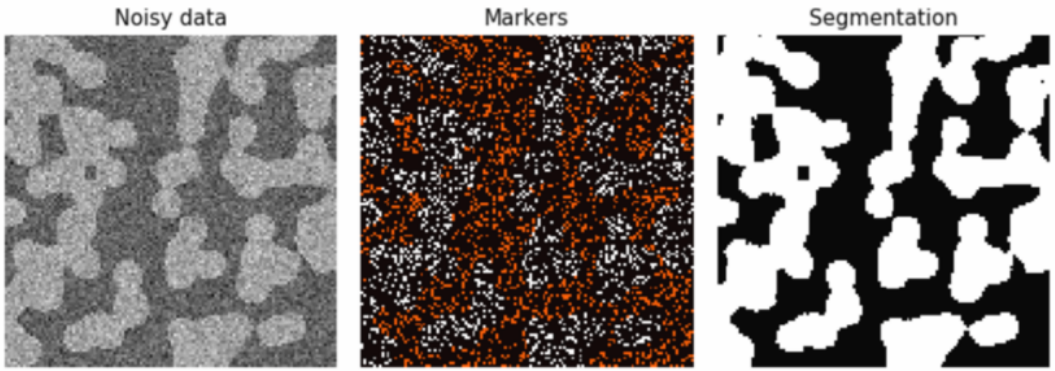
Code, figure and mini-tutorial

Random walker segmentation

The random walker algorithm [1] determines the segmentation of an image from a set of markers labeling several phases (2 or more). An anisotropic diffusion equation is solved with tracers initiated at the markers' position. The local diffusivity coefficient is greater if neighboring pixels have similar values, so that diffusion is difficult across high gradients. The label of each unknown pixel is attributed to the label of the known marker that has the highest probability to be reached first during this diffusion process.

In this example, two phases are clearly visible, but the data are too noisy to perform the segmentation from the histogram only. We determine markers of the two phases from the extreme tails of the histogram of gray values, and use the random walker for the segmentation.

[1] *Random walks for image segmentation*, Leo Grady, IEEE Trans. Pattern Anal. Mach. Intell. 2006 Nov; 28(11):1768-83

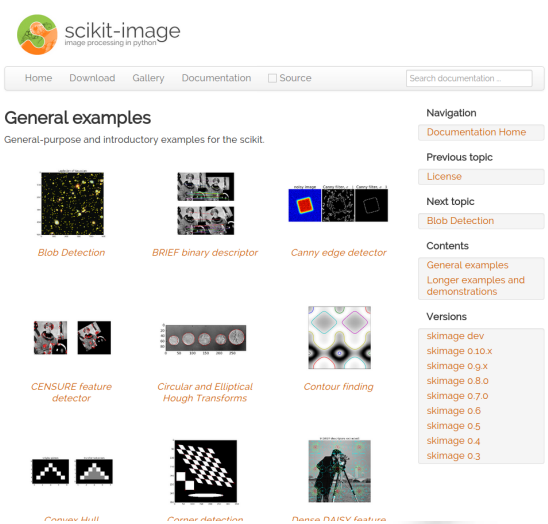


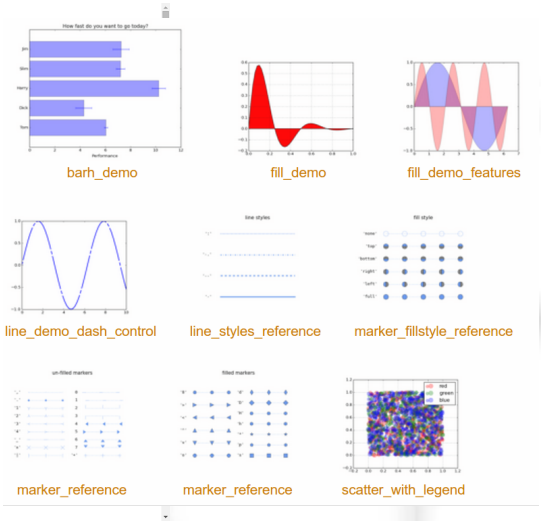

```
import numpy as np
import matplotlib.pyplot as plt

from skimage.segmentation import random_walker
from skimage.data import binary_blobs
import skimage

# Generate noisy synthetic data
data = skimage.img_as_float(binary_blobs(length=128, seed=1))
data += 0.35 * np.random.randn(*data.shape)
markers = np.zeros(data.shape, dtype=np.uint)
markers[data < -0.3] = 1
markers[data > 1.3] = 2

# Run random walker algorithm
labels = random_walker(data, markers, beta=10, mode='bf')

# Plot results
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(8, 3.2),
                                    sharex=True, sharey=True)
ax1.imshow(data, cmap='gray', interpolation='nearest')
ax1.axis('off')
ax1.set_adjustable('box-forced')
ax1.set_title('Noisy data')
ax2.imshow(markers, cmap='hot', interpolation='nearest')
ax2.axis('off')
ax2.set_adjustable('box-forced')
ax2.set_title('Markers')
ax3.imshow(labels, cmap='gray', interpolation='nearest')
ax3.axis('off')
ax3.set_adjustable('box-forced')
ax3.set_title('Segmentation')

fig.tight_layout()
plt.show()
```

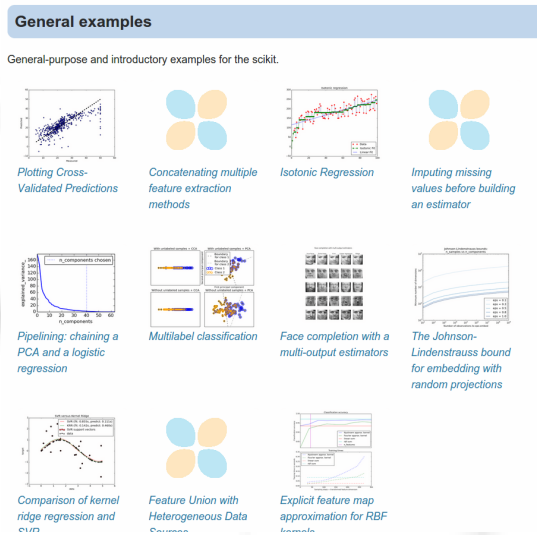
Umbrella project: sphinx-gallery

Welcome to Sphinx-Gallery's documentation!

Sphinx extension for automatic generation of an example gallery. It is extracted from the scikit-learn project and aims to be an independent general purpose extension.







Auto documenting your API with links to examples

`nilearn.image.`**`threshold_img`**(*img*, *threshold*, *mask_img=None*)

Threshold the given input image, mostly statistical or

You have gone full screen. Exit full screen (F11)

Thresholding can be done based on direct image intensities or selection threshold with given percentile.

New in version 0.2.

Parameters: **img: a 3D/4D Niimg-like object**

Image contains of statistical or atlas maps which should be thresholded.

threshold: float or str

If float, we threshold the image based on image intensities meaning voxels which have intensities greater than this value will be kept. The given value should be within the range of minimum and maximum intensity of the input image. If string, it should finish with percent sign e.g. “80%” and we threshold based on the score obtained using this percentile on the image data. The voxels which have intensities greater than this score will be kept. The given string should be within the range of “0%” to “100%”.

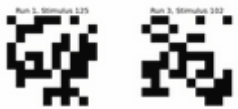
mask_img: Niimg-like object, default None, optional

Mask image applied to mask the input data. If None, no masking will be applied.

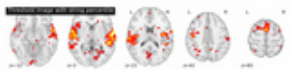
Returns: threshold_img: Nifti1Image

thresholded image of the given input image.

7.5.15.1. Examples using `nilearn.image.threshold_img`



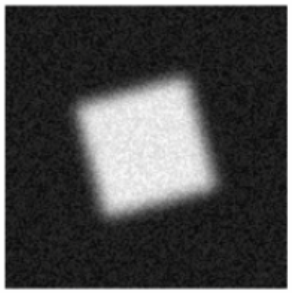
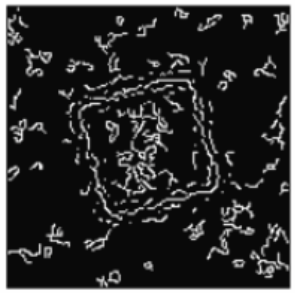
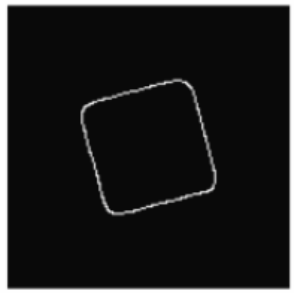
Encoding models for visual stimuli from



Region Extraction using a t-statistical map (3D)

Auto documenting your API with links to examples

noisy image | Canny filter, $\sigma = 1$ | Canny filter, $\sigma = 3$

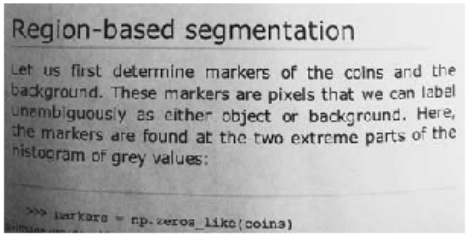




```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature


# Generate noisy image of a square
im = np.zeros((128, 128))
im[32:-32, 32:-32] = 1

im = ndi.rotate(im, 15, mode='constant')
im = ndi.gaussian_filter(im, 4)
im += 0.2 * np.random.random(im.shape)

# Compute the Canny filter for two values of sigma
edges1 = feature.canny(im)
edges2 = feature.canny(im, sigma=3)

# display results
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3),
                                    sharex=True, sharey=True)
```

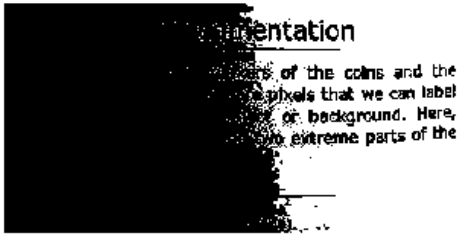
Towards a more interactive documentation?

Learning by yourself

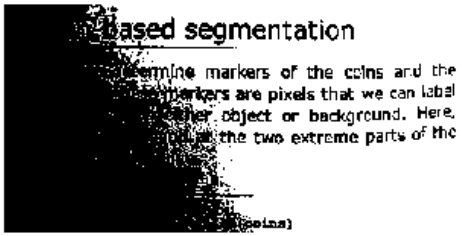
Original



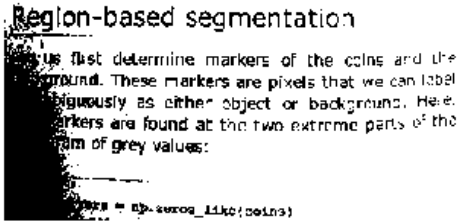
Triangle



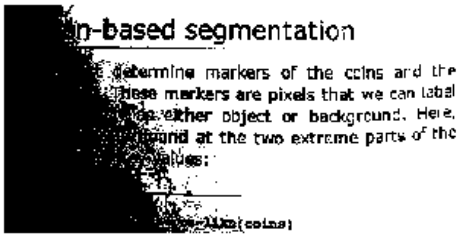
Mean



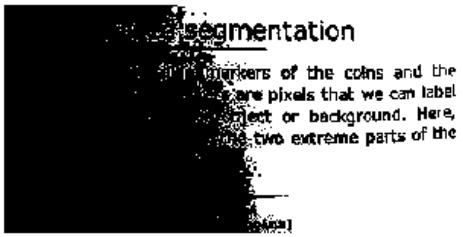
Yen



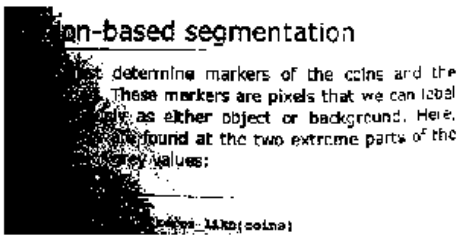
Otsu



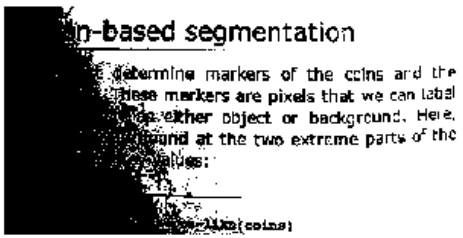
Minimum



Li

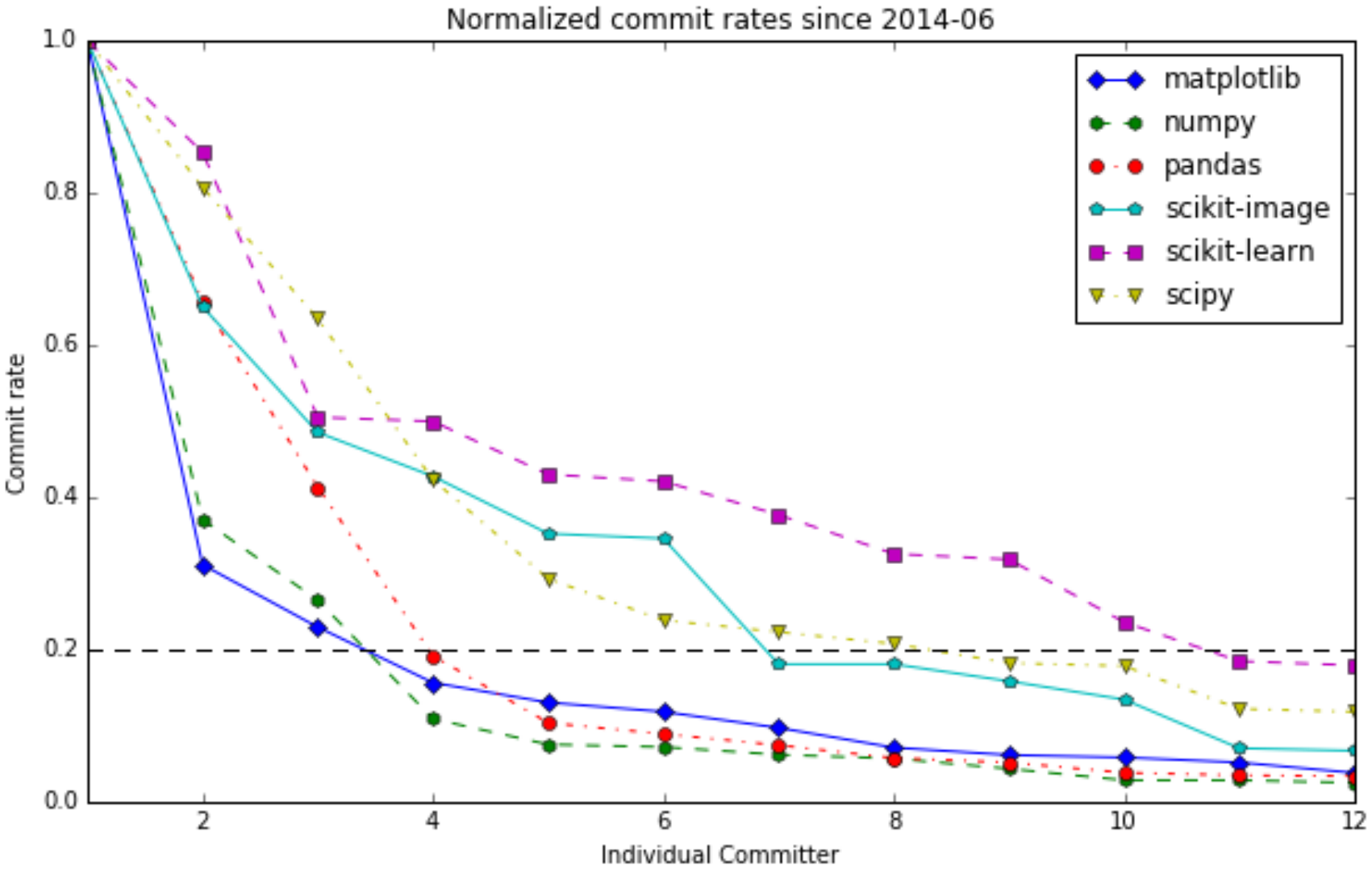


Isodata



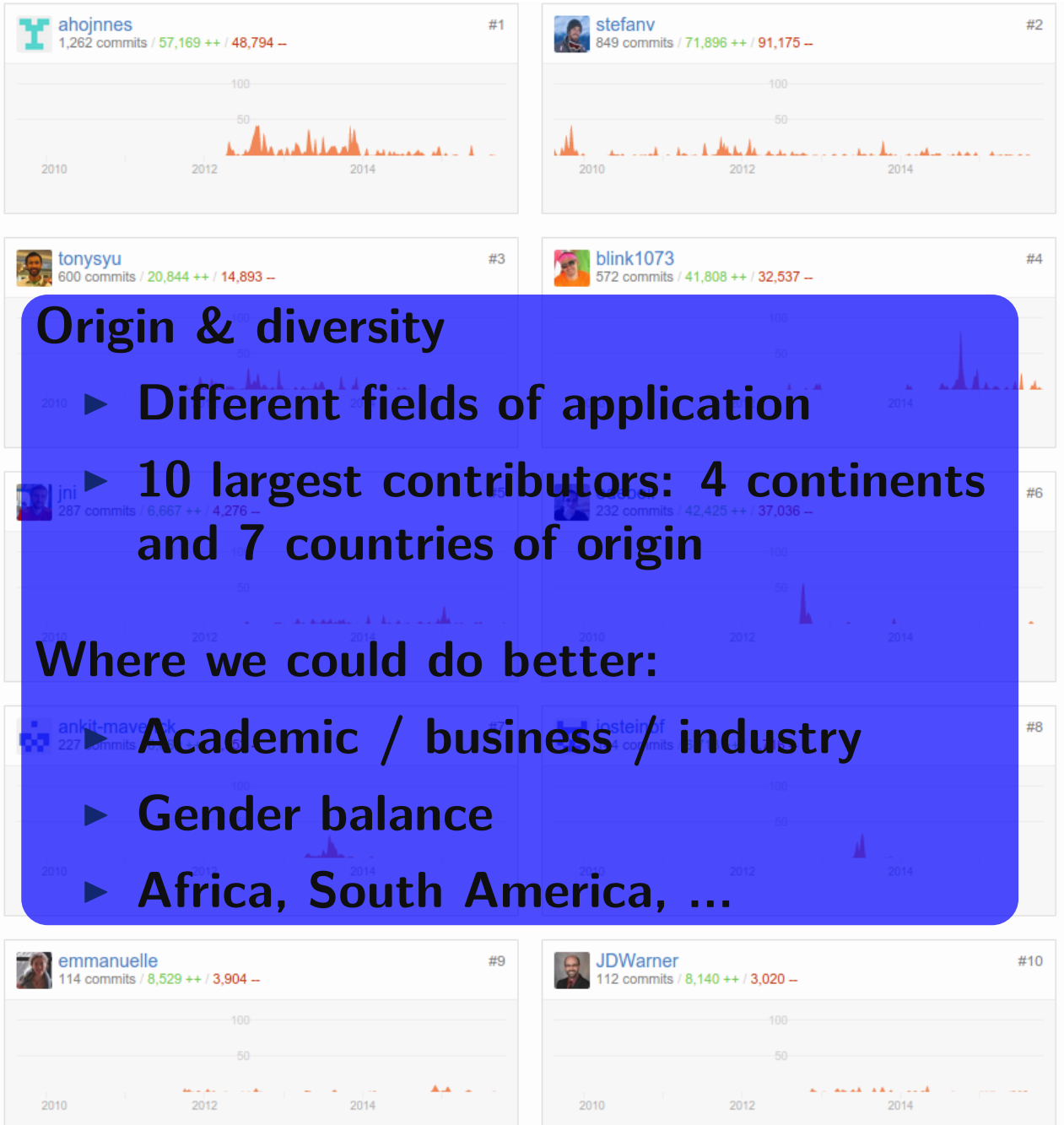
Developers and users

The people



A quite healthy curve... we can do better!

The people



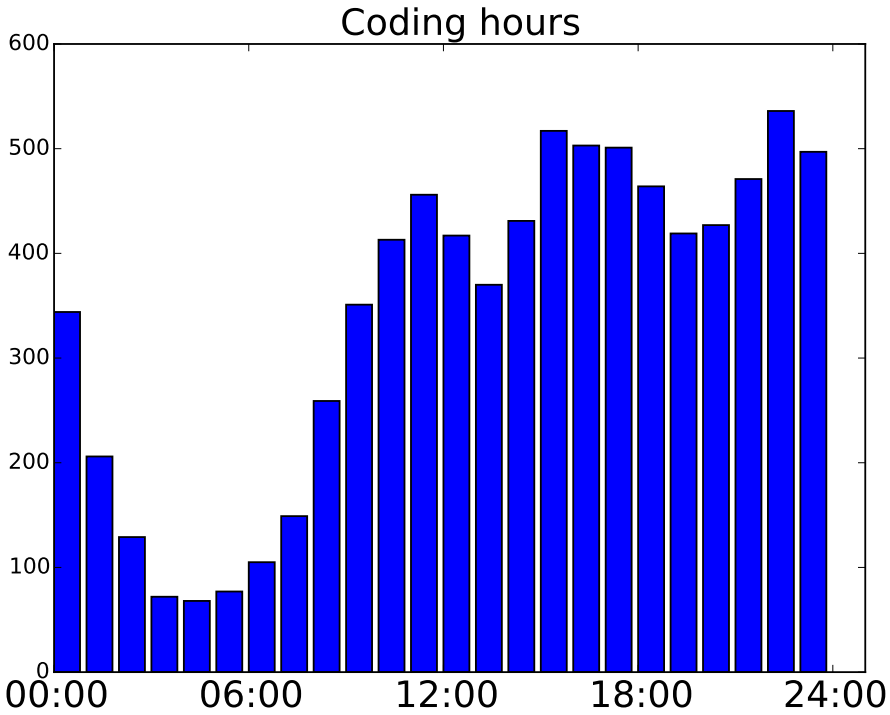
Origin & diversity

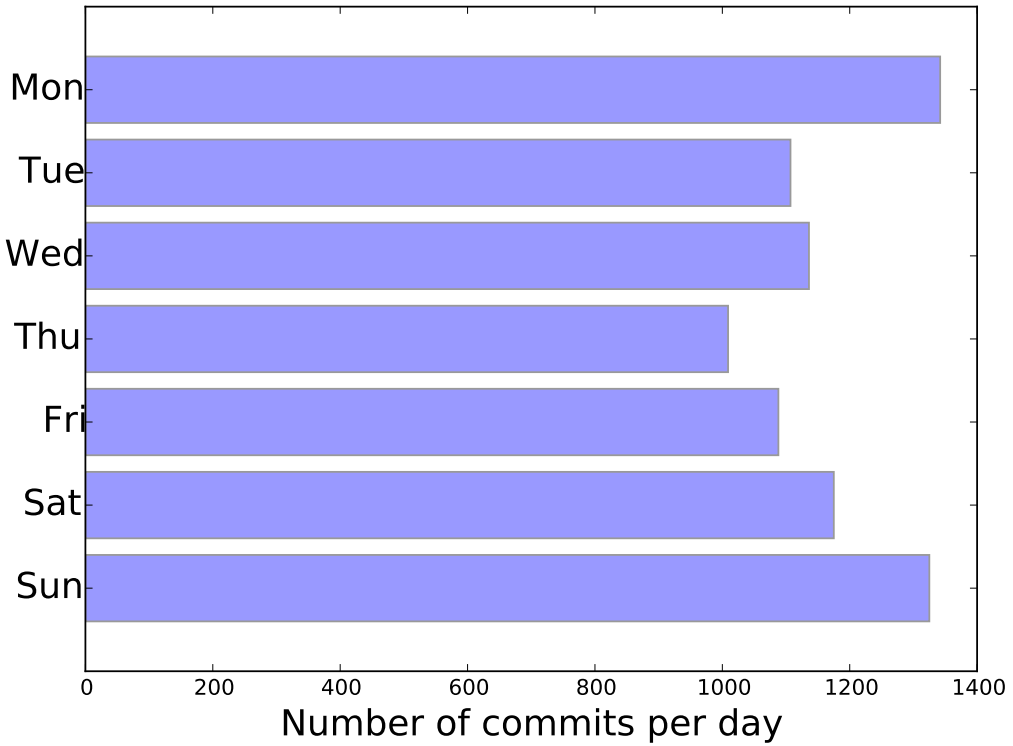
- Different fields of application
- 10 largest contributors: 4 continents and 7 countries of origin

Where we could do better:

- Academic / business / industry
- Gender balance
- Africa, South America, ...

We code when(ever) we can

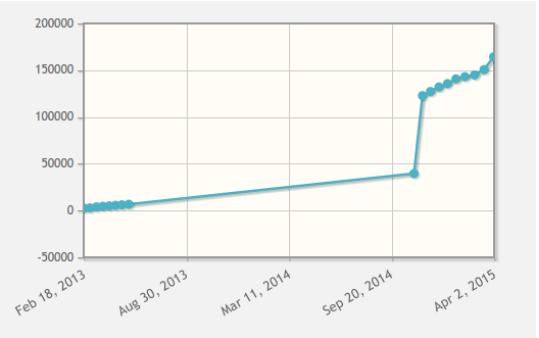




Datasheet

Package statistics

- Release 0.13 (1 - 2 release per year)
- 30000 unique visitors/month

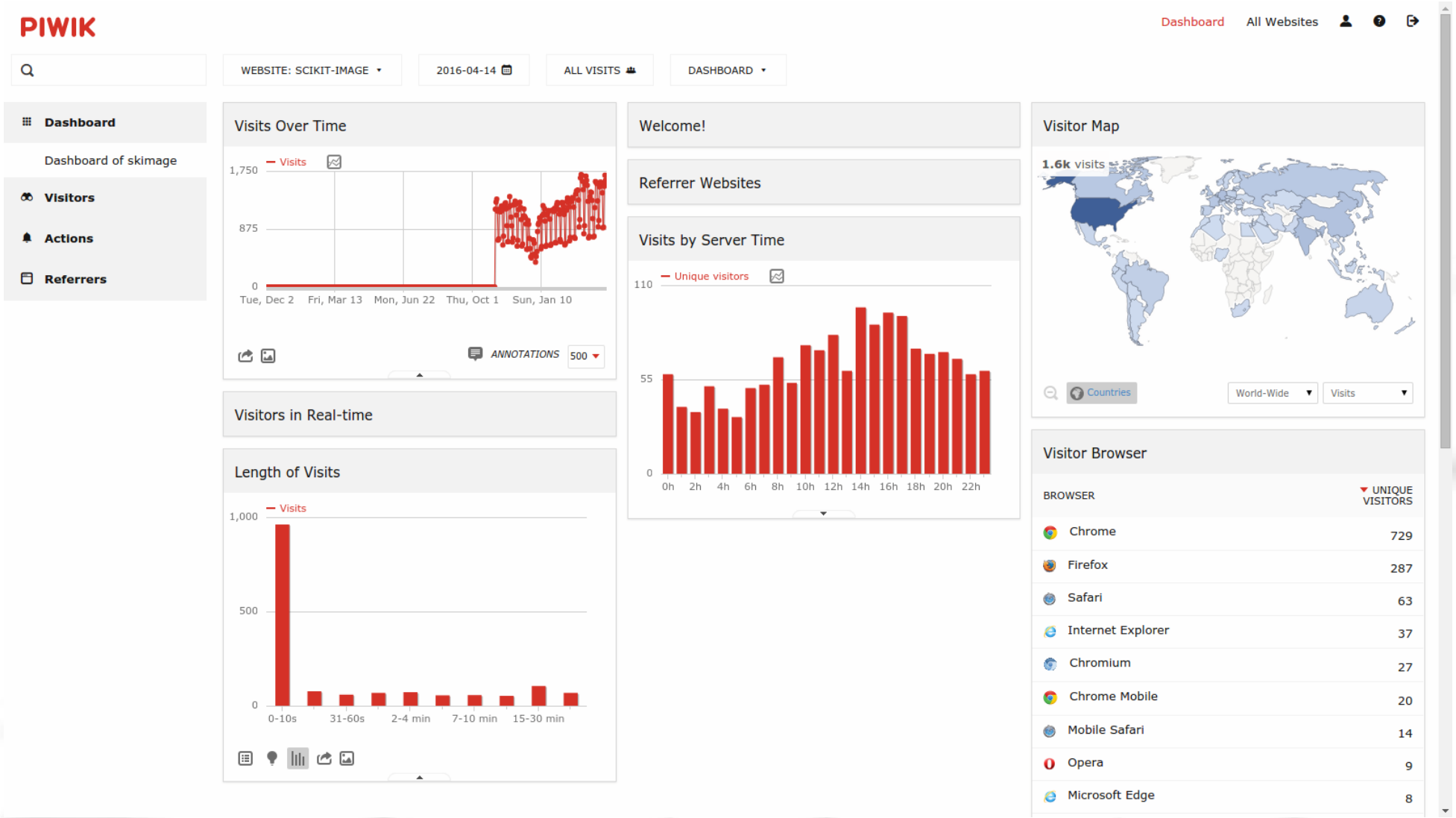


Development model

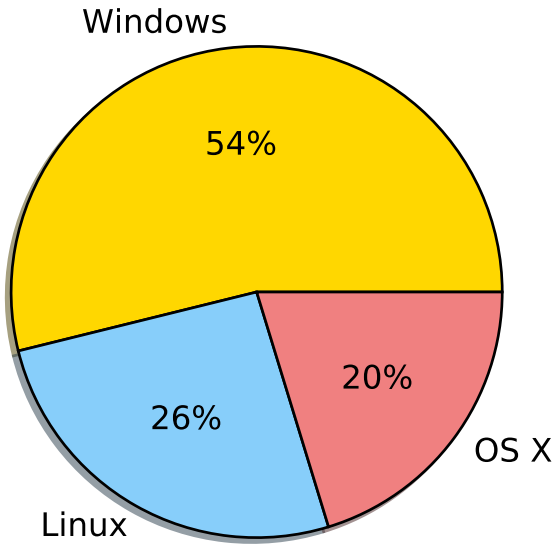
- Mature algorithms
- Only `Python` + **`Cython`** code for easier maintainability
- Focus on good practices: testing, documentation, version control
- Hosted on GitHub: thorough code review + continuous integration
- Core team of $5 - 10$ persons (close to applications)
- $>$ 200 contributors, $>$ 80% PR accepted

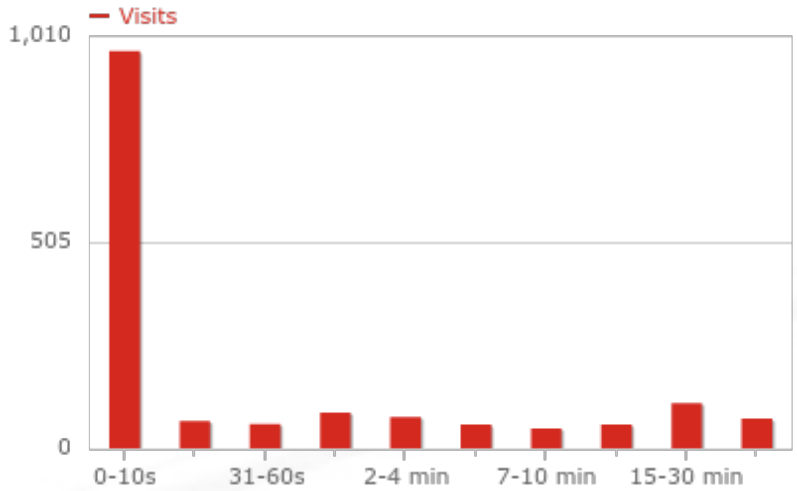
Who is your typical user?

Who is your typical user?



Who is your typical user?

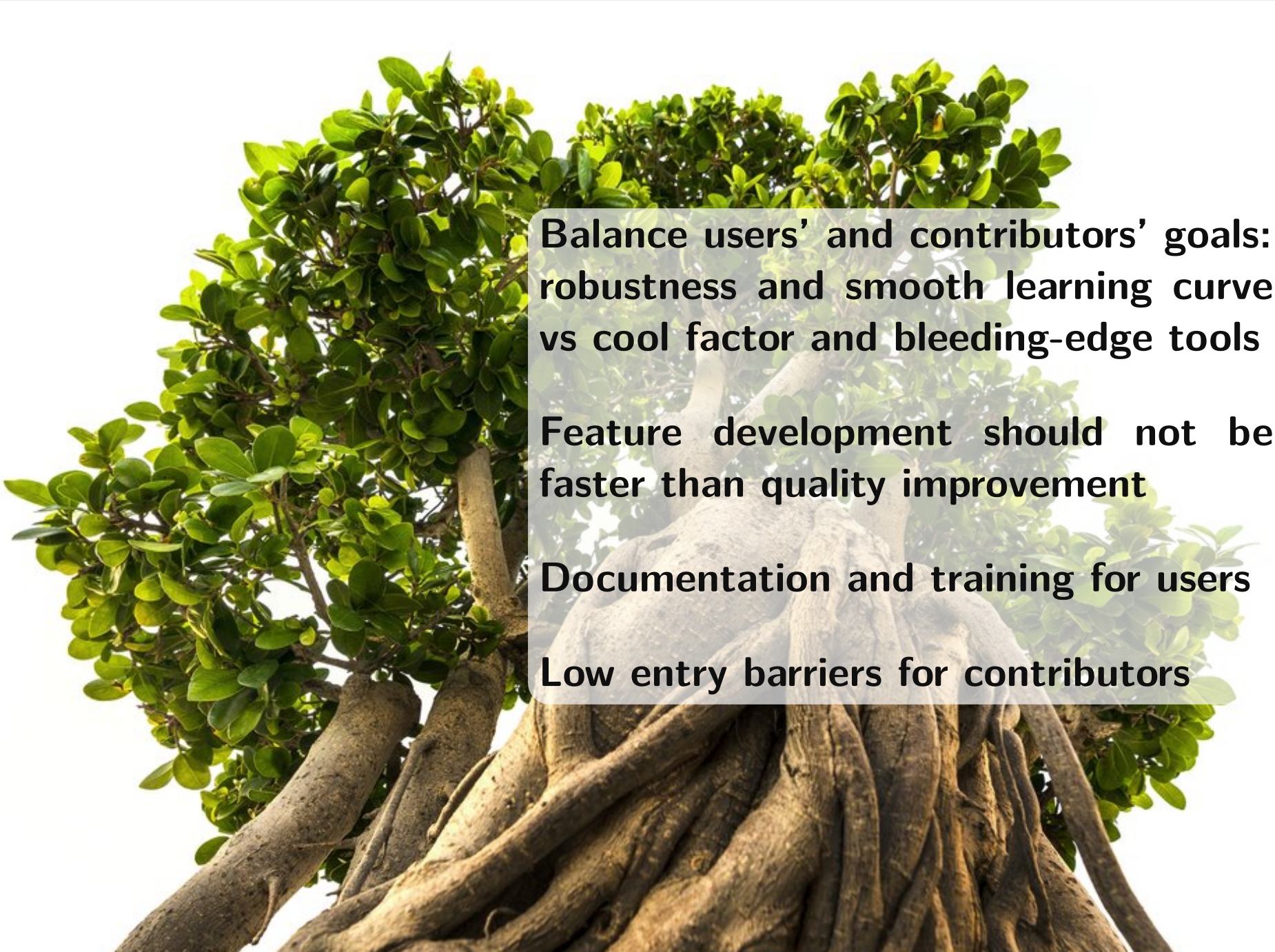




- Not a lot of hardcore geeks
- Not a lot of time on her plate
- Learning / finding information is hard

Challenges for the future

Achieving a sustainable growth



Balance users' and contributors' goals: robustness and smooth learning curve vs cool factor and bleeding-edge tools

Feature development should not be faster than quality improvement

Documentation and training for users

Low entry barriers for contributors

Massive data processing and parallelization

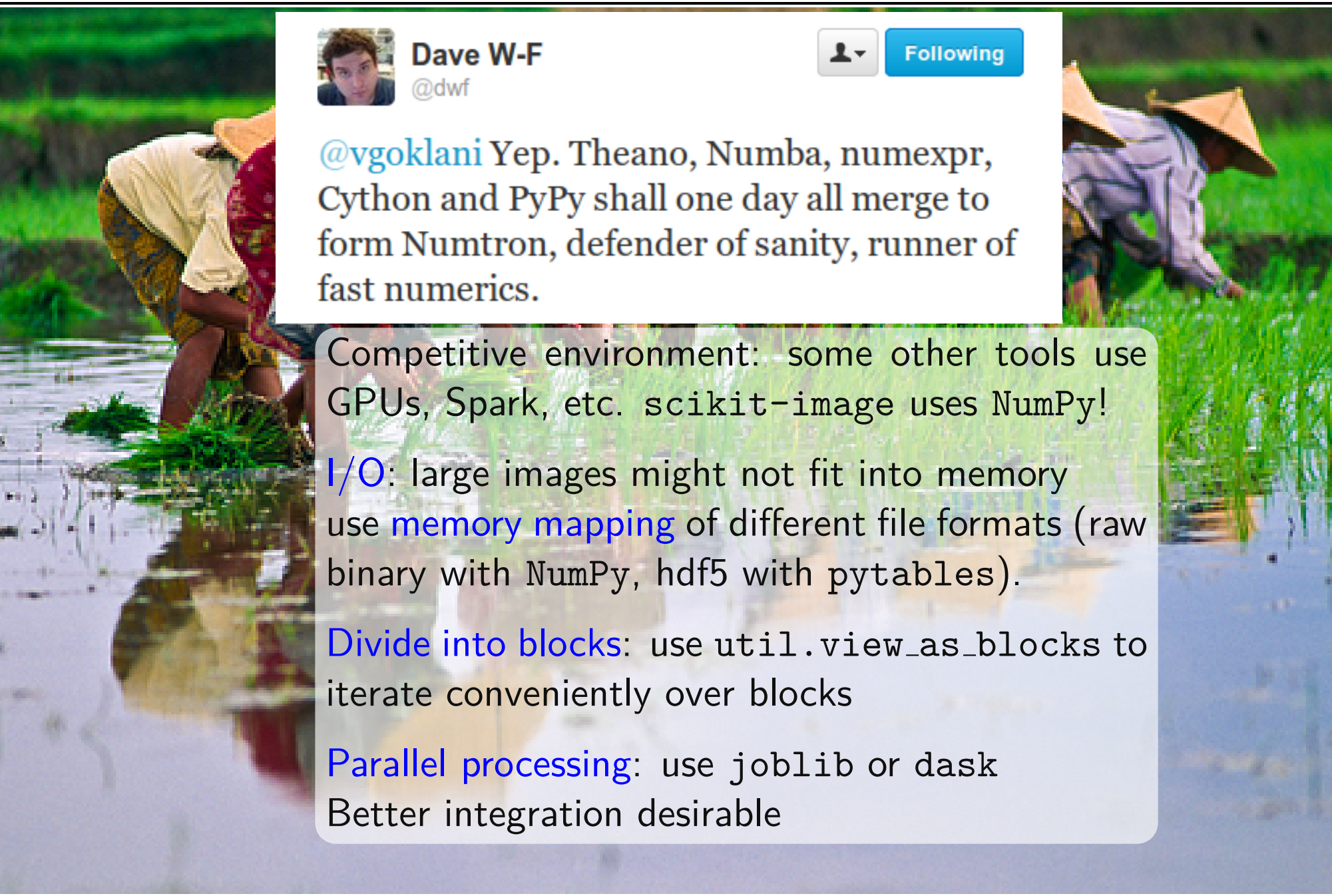
Competitive environment: some other tools use GPUs, Spark, etc. `scikit-image` uses `NumPy`!

I/O: large images might not fit into memory
use memory mapping of different file formats (raw binary with `NumPy`, hdf5 with `pytables`).

Divide into blocks: use `util.view_as_blocks` to iterate conveniently over blocks

Parallel processing: use `joblib` or `dask`
Better integration desirable

Massive data processing and parallelization



Dave W-F
@dwf

Following

@vgoklani Yep. Theano, Numba, numexpr, Cython and PyPy shall one day all merge to form Numtron, defender of sanity, runner of fast numerics.

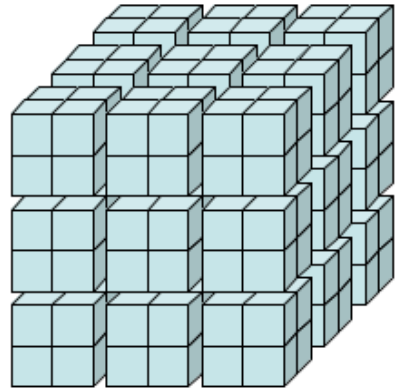
Competitive environment: some other tools use GPUs, Spark, etc. `scikit-image` uses `NumPy`!

I/O: large images might not fit into memory
use memory mapping of different file formats (raw binary with `NumPy`, hdf5 with `pytables`).

Divide into blocks: use `util.view_as_blocks` to iterate conveniently over blocks

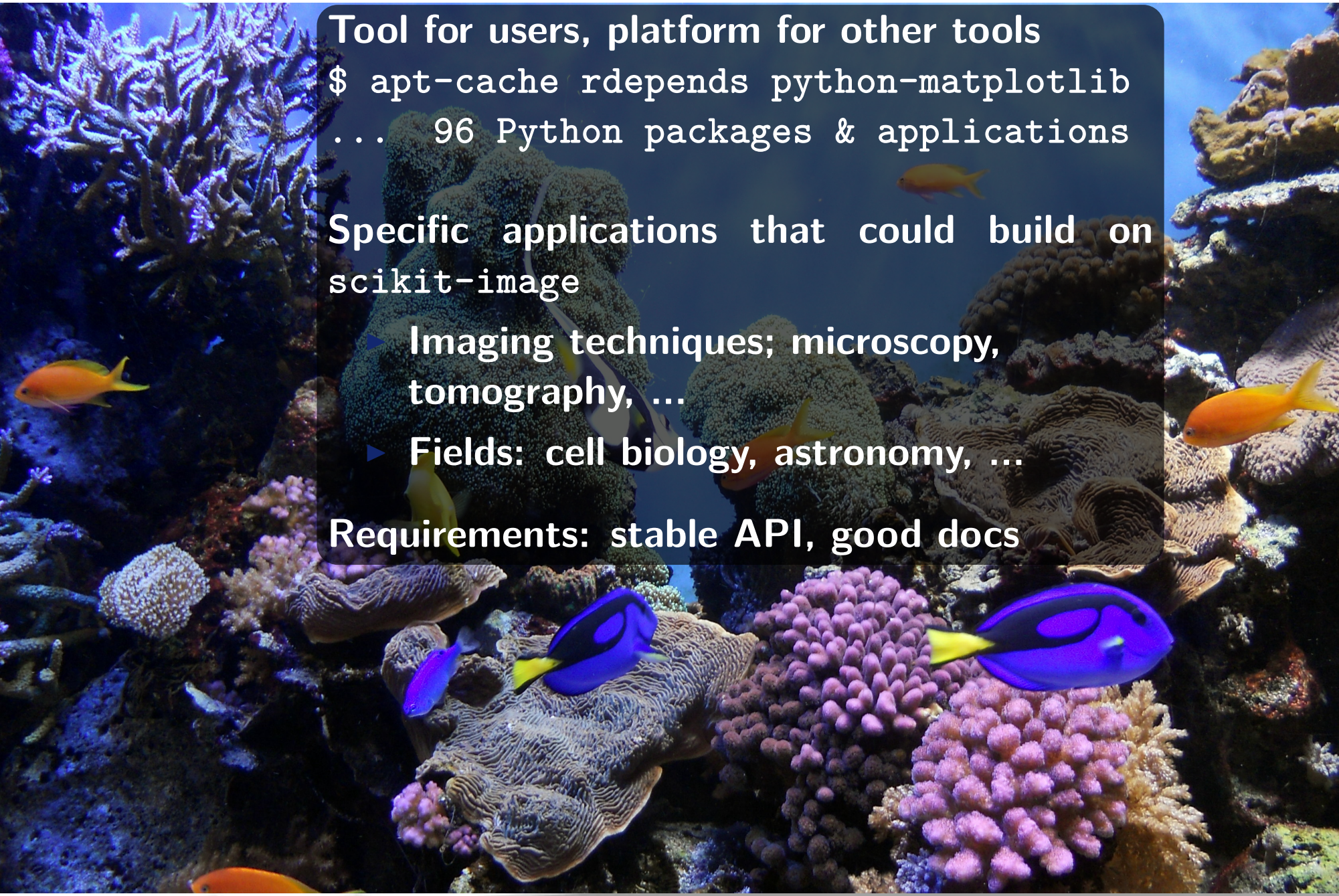
Parallel processing: use `joblib` or `dask`
Better integration desirable

Chunking data + easy parallel computation



```
>>> from skimage import util, data
>>> im = data.camera()
>>> im.shape
(512, 512)
>>> # chunks of size 134 with 8-pixel overlap
>>> chunks = util.view_as_windows(im, (134, 134), step
    =126)
>>> chunks.shape
(4, 4, 134, 134)

>>> from joblib import Parallel, delayed
>>> filtered_chunks =
Parallel(n_jobs=4)(delayed(filters.gaussian)(chunks[i,
     j]) for i in
range(4) for j in range(4))

>>> filtered_im = util.apply_parallel(filters.gaussian
    , im, depth=8)
```

A platform to build an ecosystem upon



Tool for users, platform for other tools

```
$ apt-cache rdepends python-matplotlib
...   96 Python packages & applications
```

Specific applications that could build on `scikit-image`

- Imaging techniques; microscopy, tomography, ...
- Fields: cell biology, astronomy, ...

Requirements: stable API, good docs

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

Let's talk about `scikit-image` `@EGouillart`

Wanted: application-specific examples for the gallery