THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Skimage: Basic Zoom and Crop

Adam Hughes

November 26, 2013

Contents

[1]George Washington University; Dept. of Physics; Condensed Matter Group (PI: Mark Reeves-reevesme@gwu.edu). Content claim excludes all references to to open source software, including iPython and its corresponding notebook and nbconvert utilities. Please direct inquiries to Adam Hughes (hugadams@gwmail.gwu.edu)

1 Cropping and Zooming functions

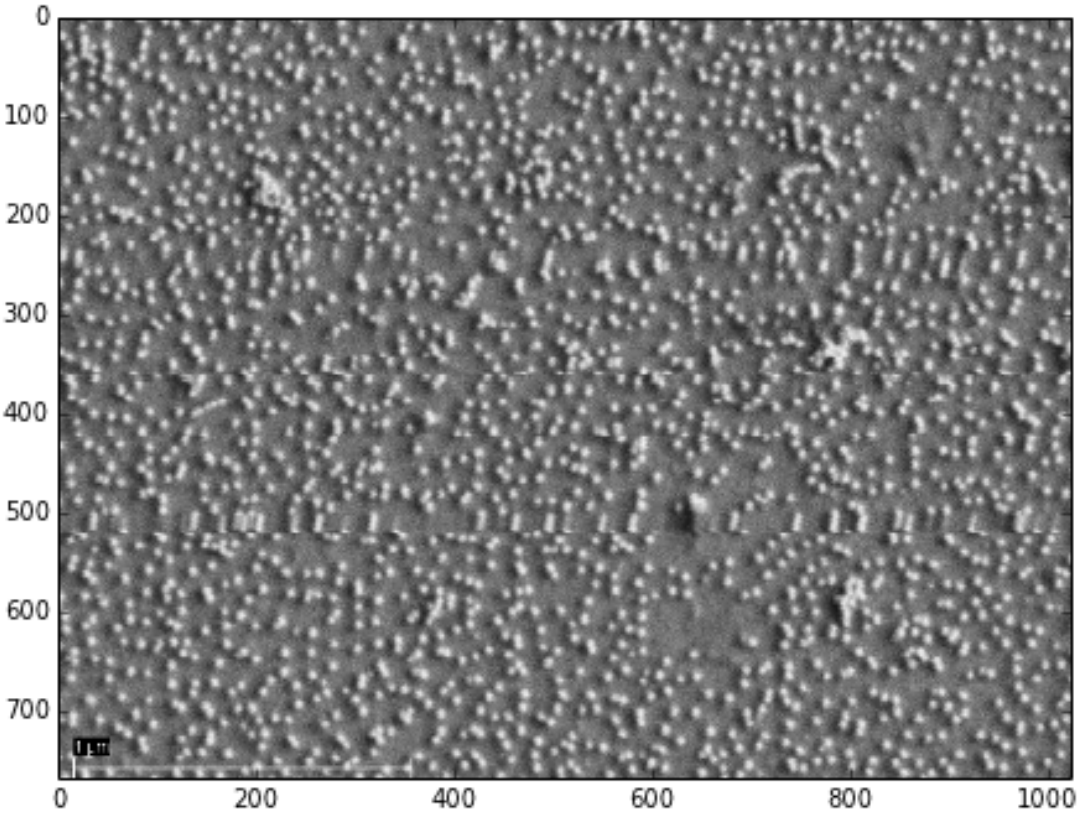
1.1 Imports; plot settings; import image

Load a scanning electron microscope example image (1024x768) of 22-nm gold nanoparticles on a silanized glass optical fiber.

In [13]:
```
plt.rcParams['figure.figsize'] = 8, 5.5
import numpy as np
from skimage.io import imread
```

In [14]:
```
image = imread('images/APRIL_29_f2_100000.tif')
imshow(image, plt.cm.gray)
```

Out [14]: <matplotlib.image.AxesImage at 0x47fe8d0>



1.2 Define three new functions: crop, zoom, zoomplot

crop(): merely slices the image

zoom(): crops the image and plots it; wraps imshow()

zoomplot(): plots zoomed image and full image side-by-side

In [15]:
```
def crop(image, xi, yi, xf, yf, *imshowargs, **imshowkwds):
''' Crop a rectangle; coordinates 0,0 start at upper left corner.
        If bounds exceed image limits, raises Error.
        Allows for xf/yf > xi/yi for more flexible rectangle drawing.'''

img_xf, img_yf = image.shape

for x in (xi, xf):
if x < 0 or x > img_xf:
raise AttributeError('Cropping bounds (%s, %s) exceed'
' image horizontal range (%s, %s)' % (xi, xf, 0, img_xf))

for y in (yi, yf):
if y < 0 or y > img_yf:
raise AttributeError('Cropping bounds (%s, %s) exceed'
' image vertical range (%s, %s)' % (yi, yf, 0, img_yf))

# Reverse bounds if final exceeds initial
if yf < yi:
yi, yf = yf, yi

if xf < xi:
xi, xf = xf, xi

image = image[yi:yf, xi:xf]
return image
```

In [16]:
```
def zoom(image, xi, yi, xf, yf, *imshowargs, **imshowkwds):
''' Plot zoomed-in region of rectangularly cropped image'''

cutimage = crop(image, xi, yi, xf, yf)
return imshow(cutimage, *imshowargs, **imshowkwds)
```

In [17]:
```
from __future__ import division
def zoomplot(image, xi, yi, xf, yf, *imshowargs, **imshowkwds):
''' Plot full and cropped image side-by-side.
        Draws a rectangle on full image to show zooming coordinate.
           - Special keywords "lw", "ls", "color", "orient" pertain
             to rectangle.  See examples
    '''

# Pop keywords for rectangle
lw = imshowkwds.pop('lw', '2')
ls = imshowkwds.pop('ls', '-')
color = imshowkwds.pop('color', 'y')
orient = imshowkwds.pop('orient', 'h')

if orient in ['h', 'horizontal']:
subshape = {'nrows':1, 'ncols':2}
elif orient in ['v', 'vertical']:
subshape = {'nrows':2, 'ncols':1}
else:
raise AttributeError('Plot orientation "%s" not understood' % orient)

# Normalize coordinates for axhline/axvline
img_ymax, img_xmax = image.shape
xi_norm, xf_norm = xi / img_xmax, xf / img_xmax
yi_norm, yf_norm = (img_ymax - yi) / img_ymax, \
(img_ymax - yf) / img_ymax

f, (ax_full, ax_zoomed) = plt.subplots(**subshape)

ax_full.imshow(image, *imshowargs, **imshowkwds)
ax_zoomed.imshow(crop(image, xi, yi, xf, yf), *imshowargs, **imshowkwds)
```

```
# Add rectangle
ax_full.axhline(y=yi, xmin=xi_norm, xmax=xf_norm,
linewidth=lw, color=color, ls=ls)
ax_full.axhline(y=yf, xmin=xi_norm, xmax=xf_norm,
linewidth=lw, color=color, ls=ls)
ax_full.axvline(x=xi, ymax=yi_norm, ymin=yf_norm,
linewidth=lw, color=color, ls=ls)
ax_full.axvline(x=xf, ymax=yi_norm, ymin=yf_norm,
linewidth=lw, color=color, ls=ls)

plt.show()
```

1.3 Some Examples

Cropping is quite straightforward. We will choose a rectangle with coordinates:

```
xi = 100, yi =100
xf = 500, yf =400
```

This should yield a **300 X 400** pixel image, preserving the aspect ratio (although not required).

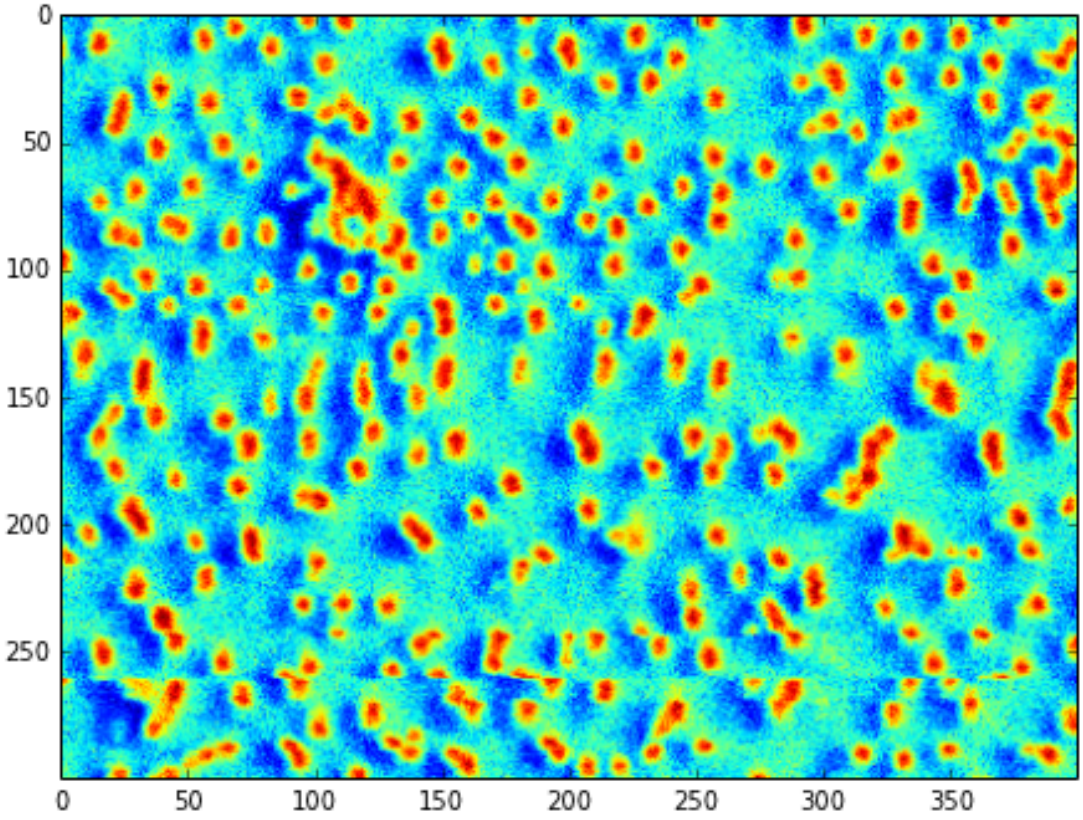
In [18]:
```
print 'Image shape:  ', image.shape
print 'Cropped shape:', crop(image, 100, 100, 500, 400).shape
```

```
Image shape:   (768, 1024)
Cropped shape: (300, 400)
```

We can use **zoom()** to crop and plot in one command:

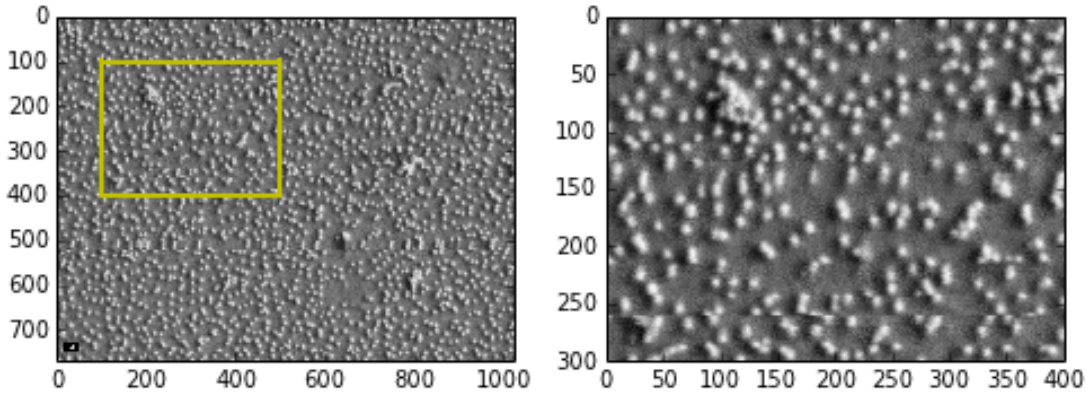
In [19]:
```
zoom(image, 100,100,500,400)
```

Out [19]: <matplotlib.image.AxesImage at 0x46ab810>


0
50
100
150
200
250
0
50
100
150
200
250
300
350

Now, let's look at the full and zoomed images side-by-side with **zoomplot()**

```
In [20]: zoomplot(image, 100,100,500,400, plt.cm.gray)
```



Change the orientation, line attributes and colormap

```
In [21]: zoomplot(image, 100,100,500,400, plt.cm.autumn,
ls='--', lw='4', color='white', orient='v')
```