COMP 364: Computer Tools for Life Sciences

Introduction to image analysis with scikit-image
(part two)

Christopher J.F. Cameron and Carlos G. Oliver
Key course information

Quiz #9
- available now (closes 11:59:59 pm)
  - covers topics from the last two weeks

HW4
- deadline extended
- due Wednesday, November 29th, 2017 at 11:59:59 pm

HW5
- available now
- due Thursday, December 7th, 2017 at 11:59:59 pm
  - shorter than previous assignments
- no extensions will be given
Outline

Today, **corner detection** using Python’s scikit-image module
- starting with a toy example (‘checkerboard’)
- then applying a detection algorithm, called ‘Harris’
- and finally explaining the Harris algorithm

Corner detection is only one aspect of image analysis
- if you’re interested, see these scikit-image tutorials: http://scikit-image.org/docs/dev/auto_examples/
To better understand corner detection

- create a checkerboard using NumPy

```python
import numpy as np

# create checkerboard
board = np.zeros((100,100),dtype=np.bool)
ind = np.arange(100).reshape((10,10))[:,::2].flatten()
# set 0,2,4,.. rows to black
board[ind,:] = True
# invert colors of blocks in 0,2,4,... columns
board[:,ind] = np.invert(board[:,ind])
# convert bools to floats
board = np.where(board,1.,0.)
```
import matplotlib.pyplot as plt

def create_image(image):
    
    """Creates image""
    fig, ax = plt.subplots()
    ax.imshow(image, cmap="Greys")
    ax.set_xlim([0, image.shape[1]])
    ax.set_ylim([image.shape[0], 0])
    # remove ticks
    ax.set_xticks([])
    ax.set_yticks([])
    # remove spines
    for spine in ["top", "bottom", "right", "left"]:
        ax.spines[spine].set_visible(False)
    plt.savefig("./../images/checkerboard.png")
    plt.close()
Many algorithms to choose from Python’s scikit-image module
  - for simplicity, we’ll choose the Harris algorithm
  - from the feature submodule
    http://scikit-image.org/docs/dev/api/skimage.feature.html

Calling the Harris algorithm’s function:

```python
1 from skimage.feature import corner_harris
2
3 print(corner_harris(board).shape())
4 # prints: (100, 100)
```
The Harris algorithm does not return corners

- returns a *response matrix* of confidence values
- the higher the value, the more likely that coordinate is a corner

To get a list of corners from the Harris algorithm

- use `corner_peaks()` to find corners in the response matrix

```python
from skimage.feature import corner_peaks

corners = corner_peaks(corner_harris(board), min_distance=2)
```

`min_distance` is similar to `sigma` from the Gaussian filter
Update the plotting function

Add an optional keyword argument for corners:

```python
def create_image_with_corners(image, corners=None):
```

Then represent detected corners as circles on the plot:

```python
if corners is not None:
    # convert coordinates to x and y lists
    y_corner, x_corner = zip(*corners)
    # plot corners with circle marker
    plt.plot(x_corner, y_corner, "ro", markersize=12)
```
def create_image_with_corners(image, corners=None):
    fig, ax = plt.subplots()
    ax.imshow(image, cmap="Greys")
    if corners is not None:
        # convert coordinates to x and y lists
        y_corner, x_corner = zip(*corners)
        # plot corners with circle marker
        plt.plot(x_corner, y_corner, 'ro', markersize=12)
        ax.set_xlim([0, image.shape[1]])
        ax.set_ylim([image.shape[0], 0])
        # remove ticks
        ax.set_xticks([])
        ax.set_yticks([])
        # remove spines
        for spine in ["top", "bottom", "right", "left"]:  
            ax.spines[spine].set_visible(False)
        plt.savefig("./../images/checkerboard_corners.png")
        plt.close()
The algorithm seems to work quite well, right?

▶ let’s try a more complicated image
import skimage.io as io
from skimage import img_as_float

# read image into memory
image = (io.imread("./../images/monkey.jpg"))
print(image[0][0])
# prints: [255 255 255]
# Harris algorithm expects 2D image
float_image = img_as_float(image)[:,:,0]
print(float_image[0][0])
# prints: 1.0
corners = corner_peaks(corner_harris(float_image),min_distance=2)
display_image_with_corners(image,corners)
display_image(float_image)
Sometimes less is more

For certain tasks like aligning images by corner detection
- it’s best when the algorithm detects prominent and constant meta features
- like the eyes, nose, or ear of the monkey

However, when running the Harris algorithm on the original image
- many leaves were (rightly) detected as corners
- in this case less detail is better
- applying Harris to the blurred image is a reasonable alternative
from skimage.color import rgb2gray
from skimage.exposure import equalize_hist
from skimage.filters import gaussian

image = (io.imread("./../images/monkey.jpg"))
gray_image = rgb2gray(image)
equalized_image = equalize_hist(gray_image)
for sigma in [0.5,0.75,1.0,1.25]:
    blurred_image = gaussian(equalized_image,sigma=sigma)
corners = corner_peaks(corner_harris(blurred_image),
    min_distance=2)
display_image_with_corners(image,corners)
display_image(blurred_image)
\( \sigma = 0.5 \)

Number of corners: 3207
\[
\sigma = 0.5
\]
\[ \text{Number of corners: } 821 \]
sigma = 0.75
sigma = 1.0

Number of corners: 62
\( \sigma = 1.0 \)
\[ \text{sigma} = 1.25 \]

Number of corners: 0

Why are we detecting less corners with increased smoothing?
\[ \text{sigma} = 1.25 \]

Less contrast between pixel intensities as smoothing is increased.
\[ \sigma = 50.0 \]
How does the Harris algorithm work?

The Harris algorithm operates by considering gradients

- What’s a gradient?
- A measured increase or decrease in magnitude of a property observed in passing from one point to another
- E.g., the difference between neighbouring pixels in an image

Gradient images display information about the rate of change from one pixel to the next

- In either the horizontal or vertical directions
- If a pixel has a much higher intensity than the pixel before it
- That pixel will have a large gradient value
Gradient images

The **Sobel filter** is a good way of producing two gradients from a grayscale image


```python
from skimage.filters import sobel_h, sobel_v

equalized_image = equalize_hist(gray_image)

# create horizontal gradient image
h_gradient = sobel_h(equalized_image)
display_image(h_gradient)

# create vertical gradient image
v_gradient = sobel_v(equalized_image)
display_image(v_gradient)
```
Horizontal gradient image
Vertical gradient image

Notice any differences?

► look at the bottom left branch
Gradient images

Given the following assumption
▶ high gradient means a high rate of change in pixel intensity

We can consider the following three cases for any given \((x, y)\) coordinate:

1. low gradient level in both the horizontal and vertical direction
2. high gradient level in either the horizontal or vertical direction ▶ but low in the other
3. high gradient level in both the horizontal and vertical direction
Gradient images #2

For case #1

▶ low gradient level in both the horizontal and vertical direction
▶ there is probably nothing interesting happening
▶ the pixel is the same as the ones around it

For case #2

▶ high gradient level in either the horizontal or vertical direction
  ▶ but low in the other
▶ probably indicates an edge (useful for edge detection)
▶ gradient corresponds to a lot of change in one direction but not the other
Gradient images #3

For case #3
  ▶ high gradient level in both the horizontal and vertical direction
  ▶ interpreted as the meeting of two edges
  ▶ certainly sounds like a corner

The Harris algorithm is one of the simplest corner detectors
  ▶ uses complicated linear algebra to detect case #3
  ▶ we’ll perform a weak approximation
  ▶ to generalize the idea of the algorithm
Harris algorithm ‘hack’

```python
h_gradient = sobel_h(equalized_image)
v_gradient = sobel_v(equalized_image)

# take coordinates of top n "largest" gradient values

top_n = 10000

h_large = zip(*np.where(h_gradient >
    np.sort(h_gradient.flatten())[-top_n]))

v_large = zip(*np.where(v_gradient >
    np.sort(v_gradient.flatten())[-top_n]))

# intersect coordinates

estimated_corners = list(set(h_large) & set(v_large))

display_image_with_corners(image,estimated_corners)
```
Harris algorithm ‘hack’ #2

Number of corners: 81
Next time in COMP 364

Finished with image analysis

▶ image I/O
▶ RGB color manipulation
▶ grayscale, histogram equalization, binarization
▶ Gaussian smoothing
▶ edge detection using the Harris algorithm
▶ gradient images and plotting corners on an image

In the next lecture

▶ interpreted vs. compiled languages