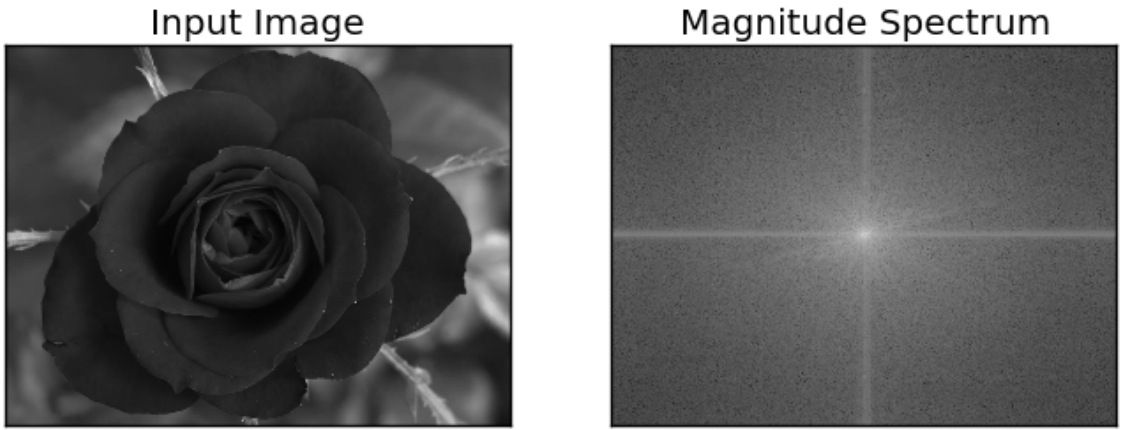
Image Processing with Python



Desert Py Meetup
26 February 2014

Sarah E. Braden

Overview

- Pillow
  - Pillow is a fork of PIL, the Python Imaging Library
  - http://python-imaging.github.io/
- OpenCV
  - http://opencv.org/
- Files for this presentation
  - https://github.com/desertpy/presentations

Why Pillow?

The Python Imaging Library, (PIL) is the library for image manipulation, however...

Why Pillow?

… PIL’s last release was in 2009

Why Pillow?

- easier to install
- supports Python 3
- active development
- actually works*

* Non-Pillow PIL really is frustrating

View on GitHub

Python Imaging

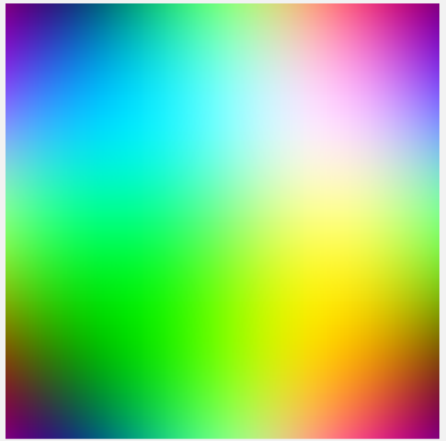
Pillow

Welcome to the Python Imaging Project

This is the home of Pillow, the popular Python Imaging Library (PIL) fork. If you have ever worried or wondered about the future of PIL: please stop! We are here to save the day.

Documentation

Our documentation which includes installation instructions, a fork of the PIL handbook and more is hosted on readthedocs.org.



Random psychedelic art made with PIL

Published with GitHub Pages

http://python-imaging.github.io/

What can Pillow do for me?

- Automatically generate thumbnails
- Apply image filters (auto-enhance)
- Apply watermarks (alpha layers)
- Extract images from animated gifs
- Extract image metadata
- Draw text for annotations (and shapes)
- Basically script things that you might do in Photoshop or GIMP for large numbers of images, in Python

Modules:

- ImageOps
- ImageMath
- ImageFilter
- ImageEnhance
- ImageStat

Pillow Setup

Pillow’s prerequisites:

https://pypi.python.org/pypi/Pillow/2.1.0#platform-specific-instructions

Warning!

Since some (most?) of Pillow's features require external libraries, prerequisites can be a little tricky

After installing prereqs:

```
$ pip install Pillow
```

Documentation

Pillow documentation:

http://pillow.readthedocs.org/en/latest/about.html

Image Basics: http://pillow.readthedocs.org/en/latest/handbook/concepts.html

List of Modules: http://pillow.readthedocs.org/en/latest/reference/index.html

Read in an image!

```
from PIL import Image

im = Image.open(infile)
im.show()

print(infile, im.format, "%dx%d" %
im.size, im.mode)
```

Important note: Opening an image file is a fast operation, independent of file size and compression. Pillow will read the file header and doesn’t decode or load raster data unless it has to.

Basic Methods

geometric transforms:

```
out = im.resize((128, 128))
out = im.rotate(45) # degrees counter-clockwise
out = im.transpose(Image.FLIP_LEFT_RIGHT)
out = im.transpose(Image.FLIP_TOP_BOTTOM)
```

crop:

```
box = (100, 100, 400, 400) #(left, upper, right, lower)
region = im.crop(box)
```

Practical Things: Make thumbnails

imgprocdemo.py

```
1  from PIL import Image
2
3  def autothumb(infile, outfile, size, format):
4      if infile != outfile:
5          try:
6              im = Image.open(infile)
7              im.thumbnail(size)
8              im.save(outfile, format)
9          except IOError:
10             print("cannot create thumbnail for", infile)
11
```

Practical Things: Apply filters

The ImageFilter module contains a number of pre-defined enhancement filters that can be used with the filter() method:

- BLUR
- CONTOUR
- DETAIL
- EDGE_ENHANCE
- EDGE_ENHANCE_MORE
- EMBOSS
- FIND_EDGES
- SMOOTH
- SMOOTH_MORE
- SHARPEN

```
from PIL import Image, ImageFilter

out1 = im.filter(ImageFilter.BLUR)
out2 = im.filter(ImageFilter.GaussianBlur(radius=20)
```

Input Image

Default Blur

Gaussian Blur, radius=20

Gaussian Blur, radius=40

Practical Things: Apply filters

UnsharpMask(*radius=2*, *percent=150*, *threshold=3*)

- radius – size of the area
- percent – % contrast change allowed in area
- threshold – minimum brightness change needed before filter takes effect

Unsharp masks basically apply a Gaussian blur to a copy of the original image and compare it to the original. If the difference is greater than a threshold setting, the images are basically subtracted.

Kernel(*size*, *kernel*, *scale=None*, *offset=0*)

- size – Kernel size, given as (width, height)
- kernel – a sequence containing kernel weights
- scale – the result for each pixel is divided by this value. Default = sum of the kernel weights
- offset – this value is added to the result, after it has been divided by the scale factor

Input Image

UnsharpMask radius=2

UnsharpMask radius=20

UnsharpMask radius=40

Practical Things: Apply watermarks

```
from PIL import Image


baseim = Image.open(imgfile)
logoim = Image.open(watermark) #transparent image
baseim.paste(logoim, (baseim.size[0]-logoim.size[0],
baseim.size[1]-logoim.size[1]), logoim)
baseim.save('new.png',"PNG")


# Important thing is the 3rd argument of the paste
function. Specifies your PNG as alpha layer so that you
avoid a black background.
```


DesertPy

What is OpenCV?

- Open source computer vision library in C++
- Includes a machine learning library to support computer vision applications
- OpenCV-Python is the Python API of OpenCV
- Large user base = good documentation and excellent online tutorials and help
- Huge library, super powerful

OpenCV Fun Facts

- In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge [1]
- You can solve sudoku puzzles with OpenCV [2]
- OpenCV is under a BSD license [3]
- OpenCV is being used to program new robots like the PR2 [4]
- Shoot squirrels [5]

[1] https://www.willowgarage.com/pages/software/opencv

[2] http://sudokugrab.blogspot.com/2009/07/how-does-it-all-work.html

[3] opencv.org

[4] http://hackermedley.org/opencv/ - interview w/OpenCV creator Gary Bradski

[5] http://www.youtube.com/watch?v=QPgqfnKG_T4

Install OpenCV for Python


```
$ sudo apt-get install python-opencv
```

Documentation: http://docs.opencv.org/modules/core/doc/intro.html

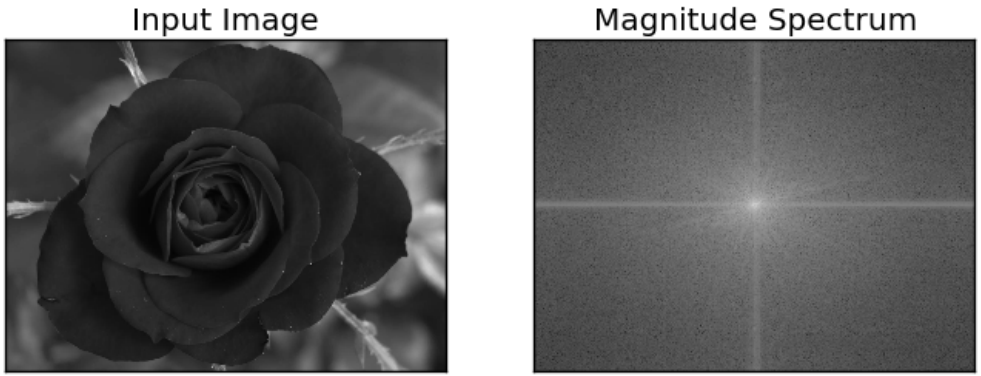
Weird things

- Color image loaded by OpenCV is in BGR mode
- But Matplotlib displays in RGB mode
- So color images will not be displayed correctly in Matplotlib if image is read with OpenCV

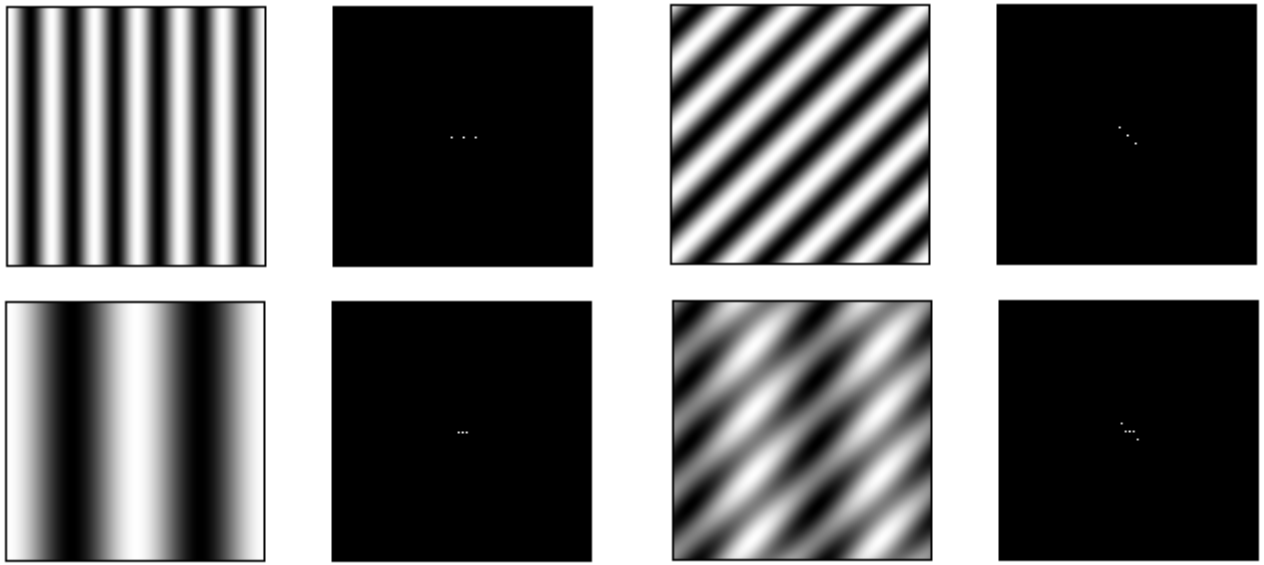
Fourier Transforms on 2D images

Use Numpy or Opencv

Center of the image represents the lower frequencies



More examples of 2D Fourier Transforms

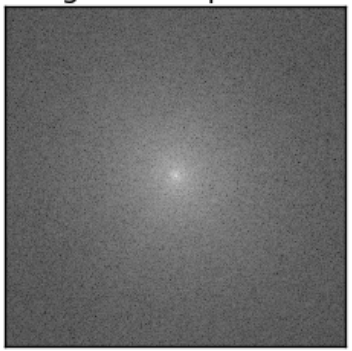


Images from: http://cns-alumni.bu.edu/~slehar/fourier/fourier.html

```python
# fast fourier transform demo

import cv2
import numpy as np
from matplotlib import pyplot as plt


# zero freqency will be in the corners
# to bring zero frequency to the center, shift the result by N/2
# in both the directions: use the function np.fft.fftshift()


def cv_fft(img):

    """
    OpenCV provides the functions cv2.dft() and cv2.idft()
    It returns the same result as numpy, but with two channels.
    First channel will have the real part of the result and
    second channel will have the imaginary part of the result.
    The input image should be converted to np.float32 first.
    """

    dft = cv2.dft(np.float32(img), flags = cv2.DFT_COMPLEX_OUTPUT)
    dft_shift = np.fft.fftshift(dft)
    a = dft_shift[:,:,0]
    b = dft_shift[:,:,1]
    magnitude_spectrum = 20*np.log(cv2.magnitude(a, b))
    return magnitude_spectrum
```

Input Image

Magnitude Spectrum



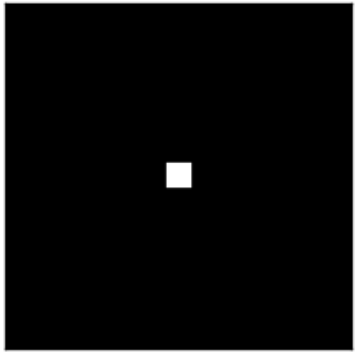
OpenCV

Mask and Inverse FFT

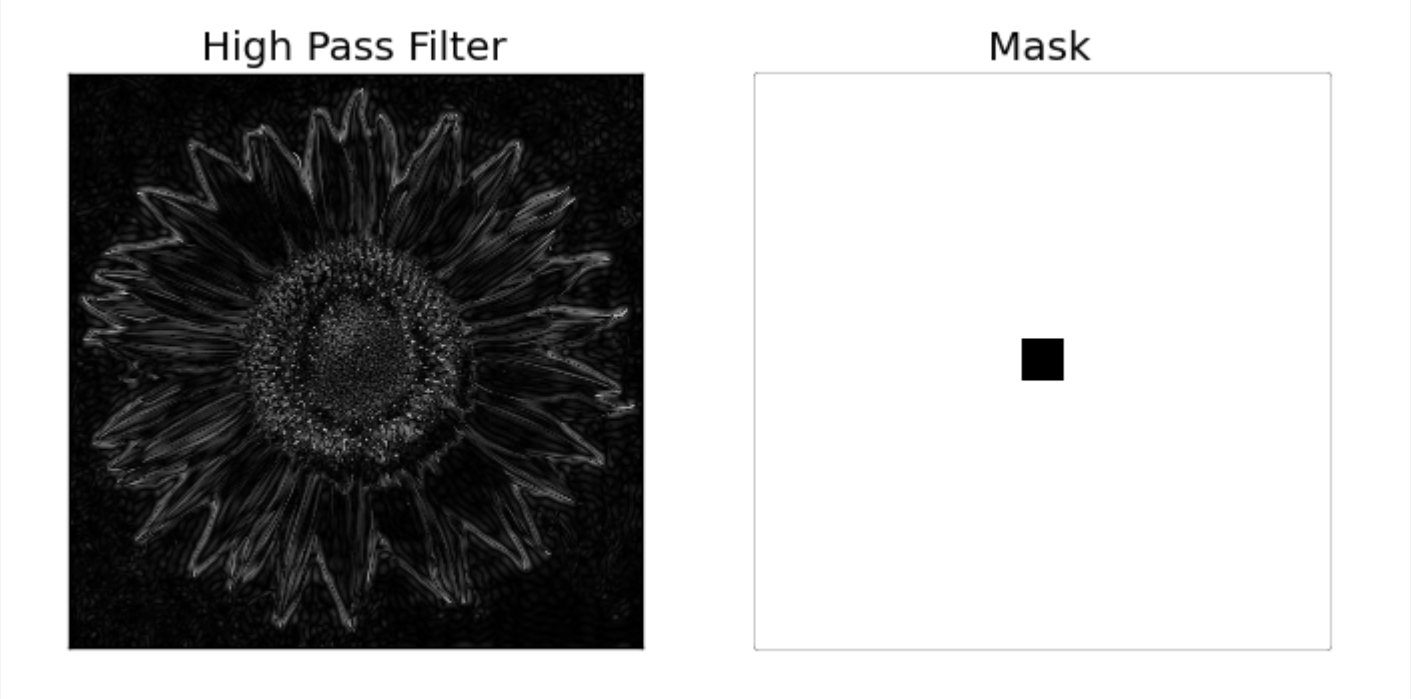
```
72
73  def create_mask(img):
74      rows, cols = img.shape
75      crow,ccol = rows/2 , cols/2
76
77      # create a mask first, center square is 1, remaining all zeros
78      mask = np.zeros((rows,cols,2),np.uint8)
79      mask[crow-30:crow+30, ccol-30:ccol+30] = 1
80
81      return mask
82
83
84  def apply_mask(mask, img):
85      # apply mask and inverse DFT
86      dft = cv2.dft(np.float32(img), flags = cv2.DFT_COMPLEX_OUTPUT)
87      dft_shift = np.fft.fftshift(dft)
88      fshift = dft_shift*mask
89      f_ishift = np.fft.ifftshift(fshift)
90      img_back = cv2.idft(f_ishift)
91      img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
92      return img_back
93
```

Inverse Image

Mask



High Pass Filter Using FFT



Hey look it’s edge-detection!

Questions?

Post questions to the DesertPy G+ page!

Sarah Braden Twitter: @ifmoonwascookie

Alternatives to Pillow?

- PythonMagickWand is an object-oriented Python interface to MagickWand based on ctypes. 21 Jan 2009?
- PythonMagick is an object-oriented Python interface to ImageMagick. Last build 22 January 2014.
- Wand is a ctypes-based ImagedMagick binding library for Python. Last release 17 June 2013.

Pillow Setup

Some (most?) of Pillow's features require external libraries.

- **libjpeg** provides JPEG functionality.
  - Pillow has been tested with libjpeg versions **6b**, **8**, and **9**
- **zlib** provides access to compressed PNGs
- **libtiff** provides group4 tiff functionality
  - Pillow has been tested with libtiff versions **3.x** and **4.0**
- **libfreetype** provides type related services
- **littlecms** provides color management
- **libwebp** provides the Webp format.
  - Pillow has been tested with version **0.1.3**, which does not read transparent webp files. Version **0.3.0** supports transparency.

Also: you may want to: sudo apt-get install imagemagick

Prerequisites are installed with on **Ubuntu 12.04 LTS** with

$ sudo apt-get install libtiff4-dev libjpeg8-dev zlib1g-dev libfreetype6-dev liblcms1-dev libwebp-dev