Image Generation, Processing & Understanding with Python

Yongduek Seo

2019-04-01

Contents

Chapter 1

Preparation & Documents

1. `python 3.x`
2. `pip install opencv-python numpy matplotlib`
3. OpenCV-Python Tutorial
4. docs.opencv.org
5. Python Image Library: Pillow
6. scikit-image.org
7. github.com/opencv

Chapter 2

Pixelwise Operations

2.1 Blending Two Images

$$I_B = \alpha I_1 + (1-\alpha) I_2, \quad \alpha \in [0,1]$$

Procedure:

1. prepare two image files
2. read the two files
3. set α to a value, e.g. 0.5
4. compute the weighted sum of the two images
5. display the result

Notice:

- `np.uint8` is the type of a pixel value for display.
- BGR in OpenCV!

Try:

- make a video showing a progressive change from one image to another by increasing α from 0 to 1 smoothly.

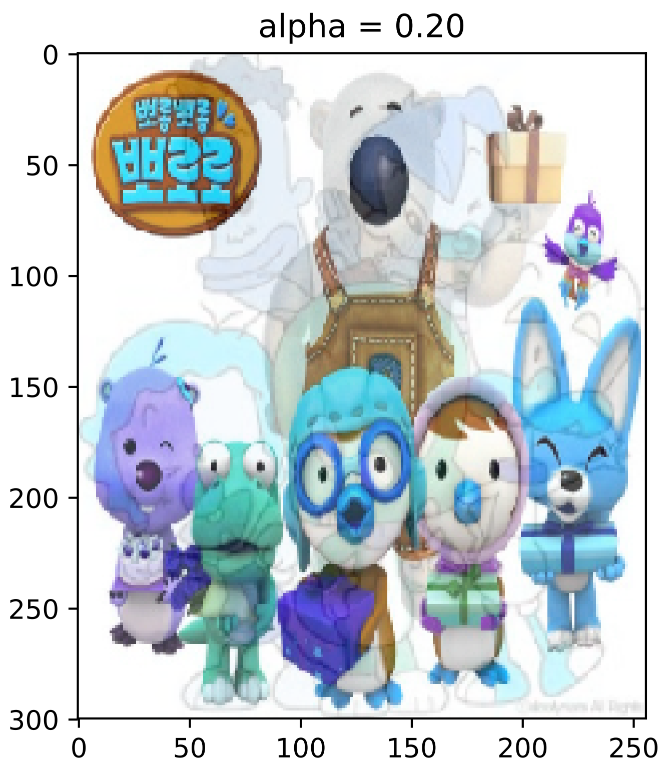
```
# filename blending.py

import sys
import numpy as np
import cv2
import matplotlib
matplotlib.use ('TkAgg')
import matplotlib.pyplot as plt


size = (256, 300)
# read two images
```

```
file1 = 'data/dooly.jpeg'
i1 = cv2.imread (file1)
if i1 is None:
    print ('image file read error: ', file1)
    sys.exit()

i1 = cv2.resize(i1, size)

file2 = 'data/pororo.jpeg'
i2 = cv2.imread (file2)
if i2 is None:
    print ('image file read error: ', file2)
    sys.exit()

i2 = cv2.resize (i2, size)

results = []
alphas = np.linspace(0, 1, 6)  # allocate alpha values
print ('alpha: ', alphas)
```

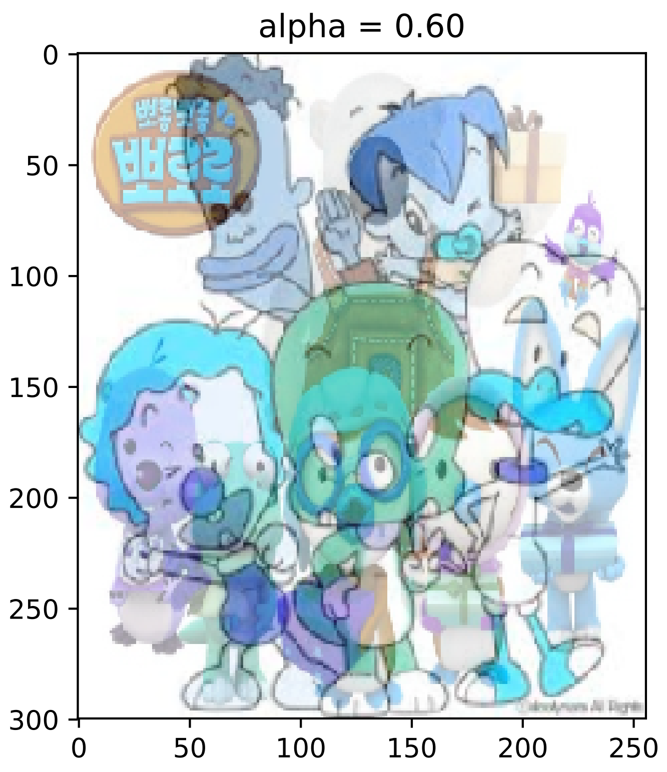
```
## alpha:  [0.  0.2 0.4 0.6 0.8 1. ]
```

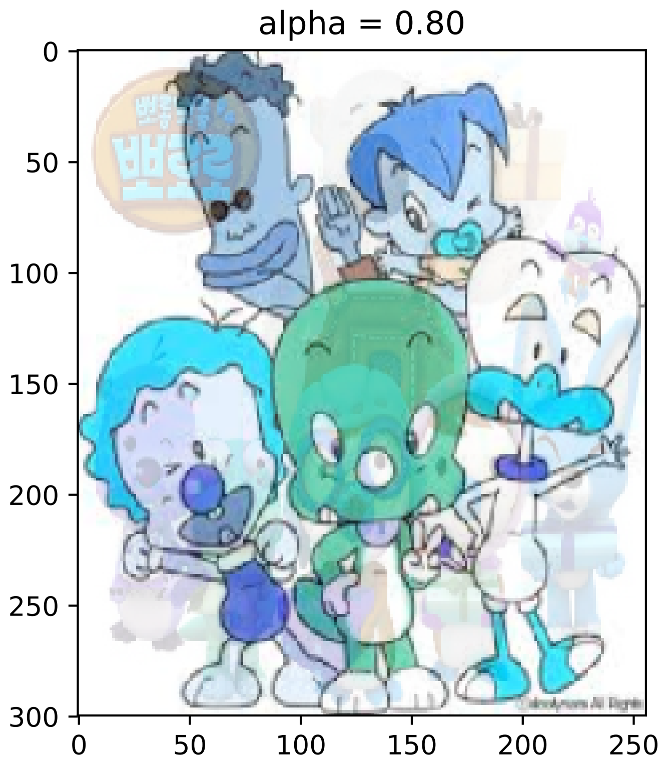
```
for a in alphas:
    J = a * i1 + (1. - a) * i2
    J = np.clip(J, 0, 255).astype(np.uint8)
    results.append (J)
    print (a)
    plt.imshow (J)
    plt.title ('alpha = %.2f' % a)
    plt.pause (1)
#
```

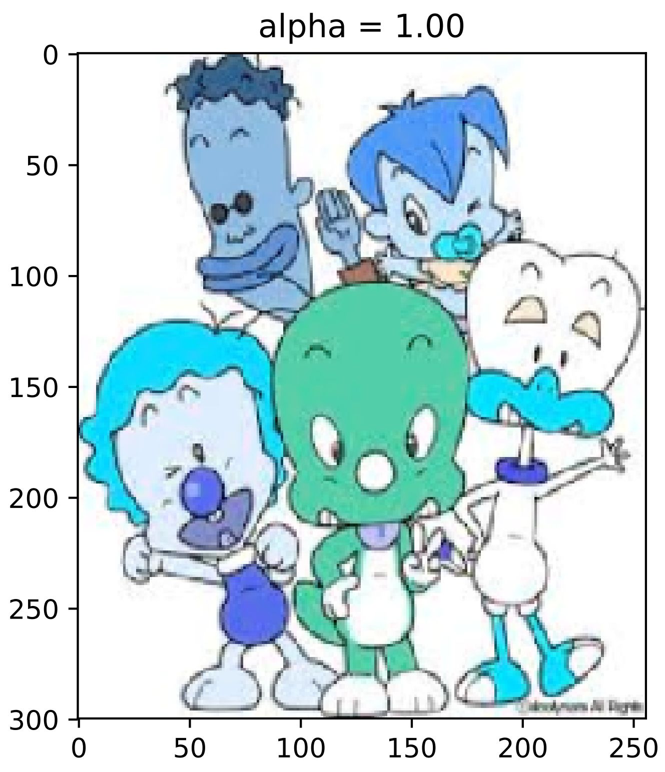
alpha = 0.00
0
50
100
150
200
250
300
0
50
100
150
200
250
뽀롱뽀롱
뽀로로


alpha = 0.20
0
50
100
150
200
250
300
0
50
100
150
200
250
뽀롱뽀롱
뽀로로

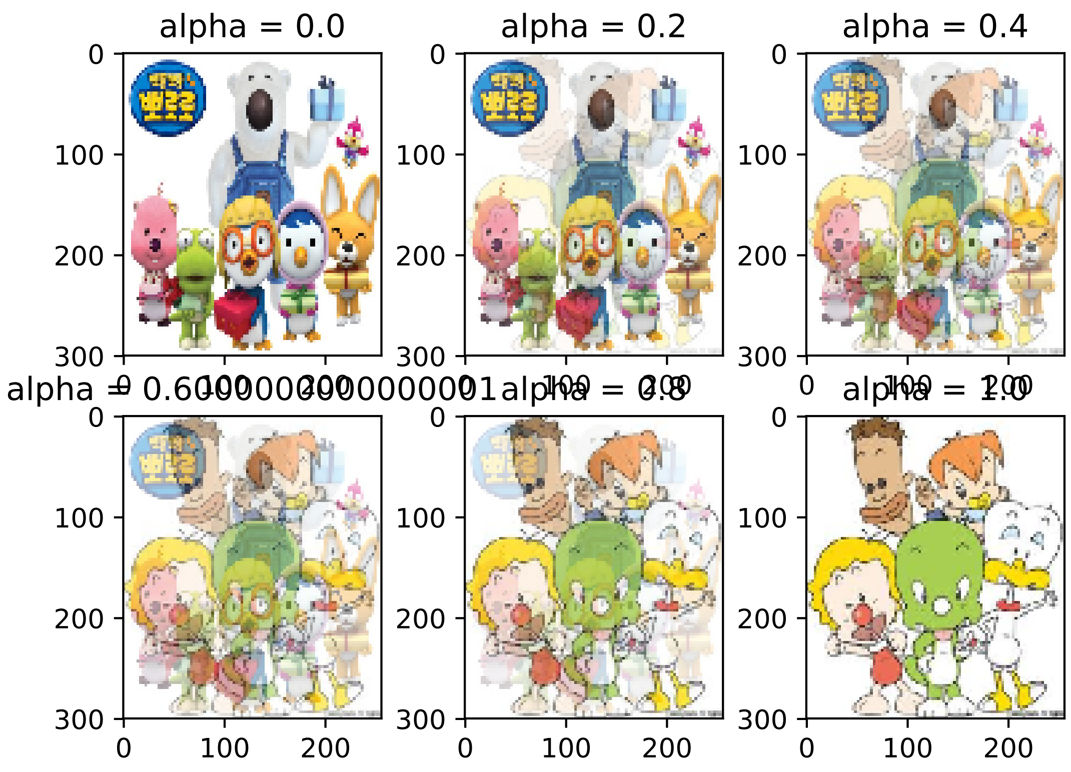
alpha = 0.40
0
50
100
150
200
250
300
0
50
100
150
200
250


alpha = 0.60
0
50
100
150
200
250
300
0
50
100
150
200
250


alpha = 0.80
0
50
100
150
200
250
300
0
50
100
150
200
250



```
fig, axes = plt.subplots (2, 3)
for i, ax in enumerate(axes.ravel()):
    ax.imshow (results[i][:,:,::-1]) # BGR -> RGB
    ax.set_title ('alpha = {}'.format(alphas[i]))
#
plt.pause (2)
```

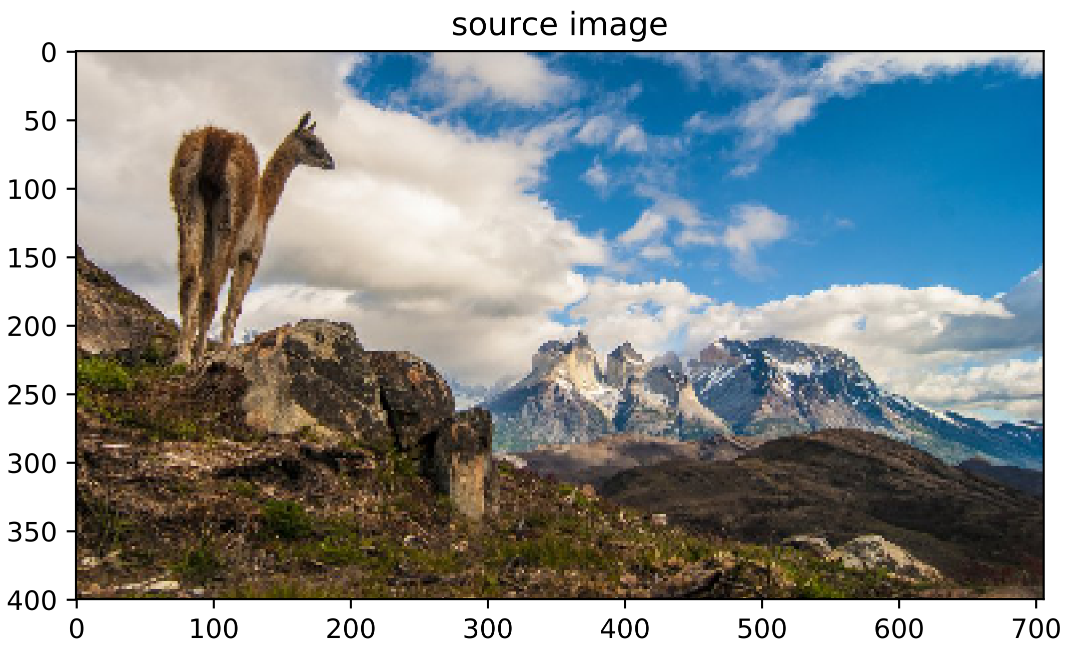


```
plt.close()

# EOF
```

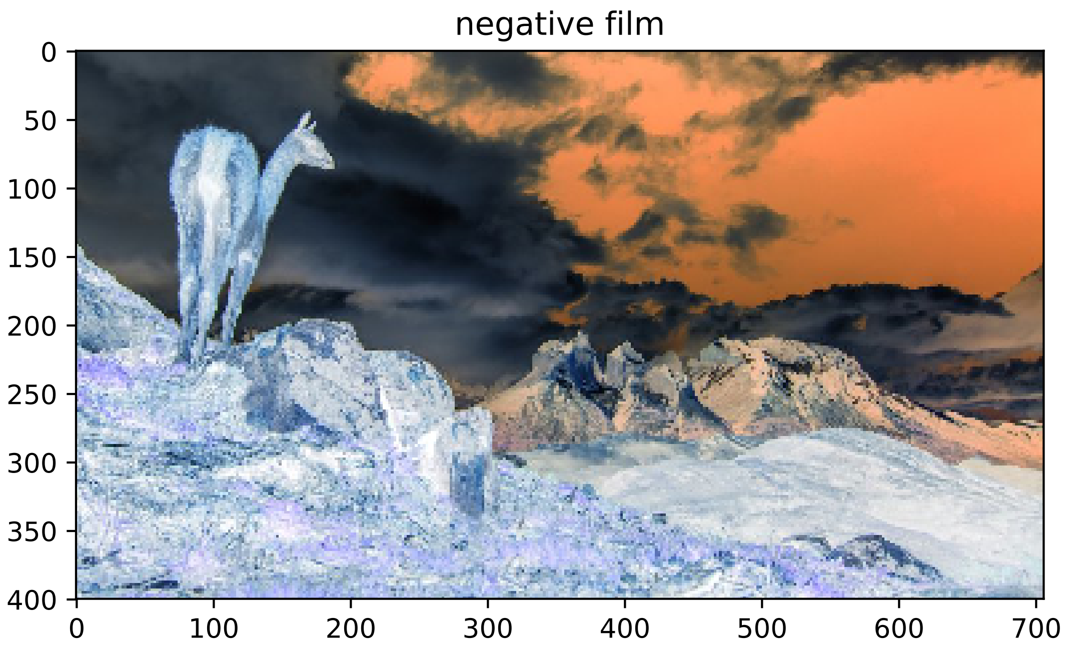
2.2 Negative Film Effect

The image looks like a negative film in old days.

```
# filename negative_film.py

import sys
import numpy as np
import cv2
import matplotlib
matplotlib.use ('TkAgg')
import matplotlib.pyplot as plt
import imageio # this will be used to load an image file

# show the image
def imshow (img, title=None):
    plt.imshow (img)
    if title is None: title = 'imshow'
    plt.title (title)
    plt.pause (1)
    plt.close ()
#

def negative_film (img):
#    return 255 - img  # simple way using numpy.
    neg = np.zeros_like (img)
    for r in range (img.shape[0]):
        for c in range (img.shape[1]):
            for d in range (img.shape[2]):
                neg[r,c,d] = 255 - img[r,c,d]
    return neg
#

img = imageio.imread ('data/torres-del-paine.jpg')

imshow (img, 'source image')
```



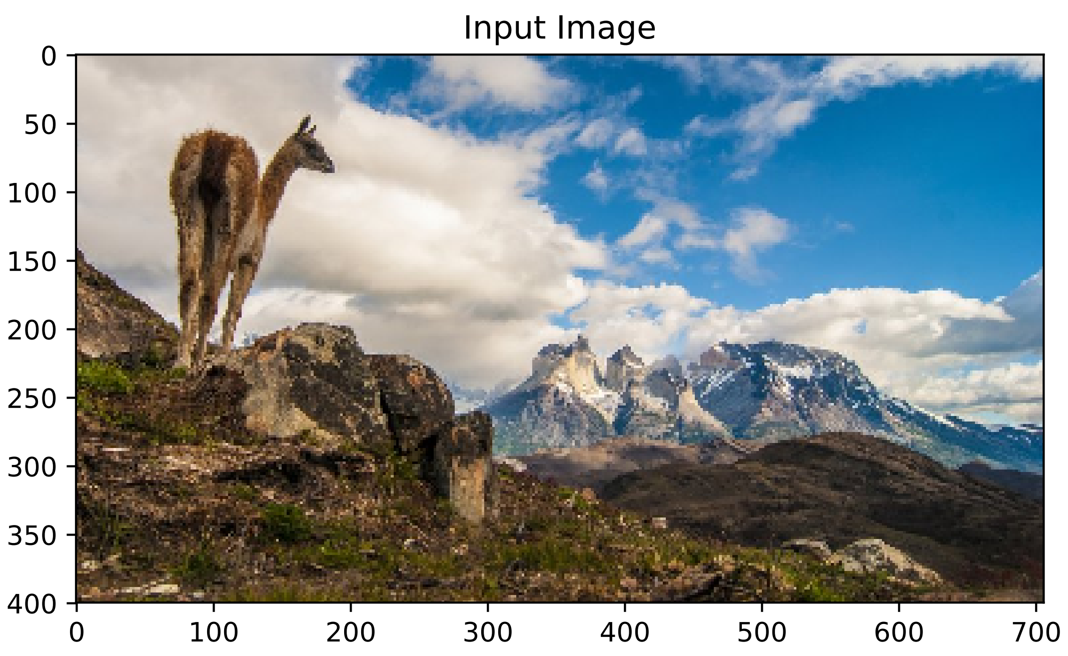
```
neg = negative_film (img)

imshow (neg, 'negative film')

# EOF
```



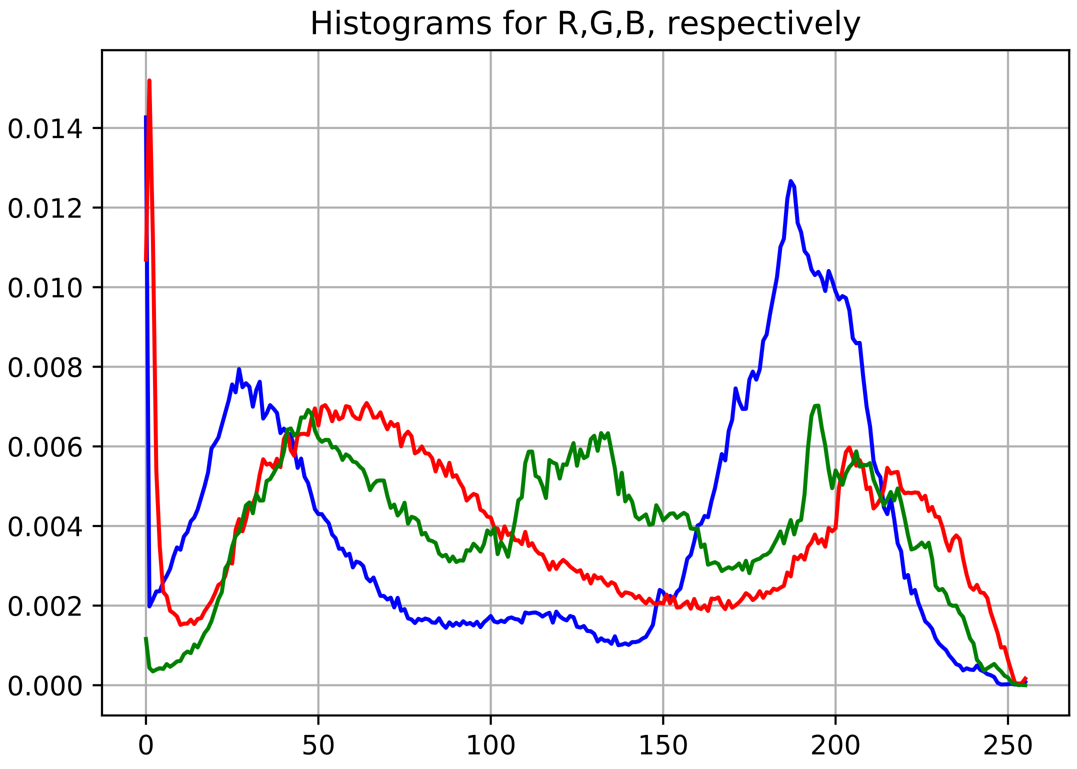
2.3 Histogram of RGB numpy image

```
# filename: numpy-hist.py

import sys
import matplotlib.pyplot as plt
import numpy as np
import cv2

imagefile = 'data/torres-del-paine.jpg'

frame = cv2.imread(imagefile)

bluehist = np.zeros((256), dtype=np.float)
redhist = np.zeros((256), dtype=np.float)
greenhist = np.zeros((256), dtype=np.float)

# make histograms, one for each color
for r in range(frame.shape[0]):
    for c in range(frame.shape[1]):
        blue_intensity = frame[r,c][0]
        bluehist[blue_intensity] += 1
        redhist[frame[r,c,2]] += 1
        greenhist[frame[r,c][1]] += 1
#

# convert to ratio = count / num_pixels
num_pixels = frame.shape[0] * frame.shape[1]
bluehist /= num_pixels
greenhist /= num_pixels
redhist /= num_pixels

plt.imshow (frame[:,:,::-1]) ## cv2's BGR -> RGB
plt.title ('Input Image')
plt.pause (1)
plt.close()

x = range(0,256,1)
plt.plot (x, bluehist, 'b', x, redhist, 'r', x, greenhist, 'g')
plt.grid(True)
plt.title ('Histograms for R,G,B, respectively')
plt.pause (1)
plt.close ()

#EOF
```

A result of the program is shown below.



```
## [<matplotlib.lines.Line2D object at 0x7f11e20e7860>, <matplotlib.lines.Line2D object at 0x7f11e20e78
```

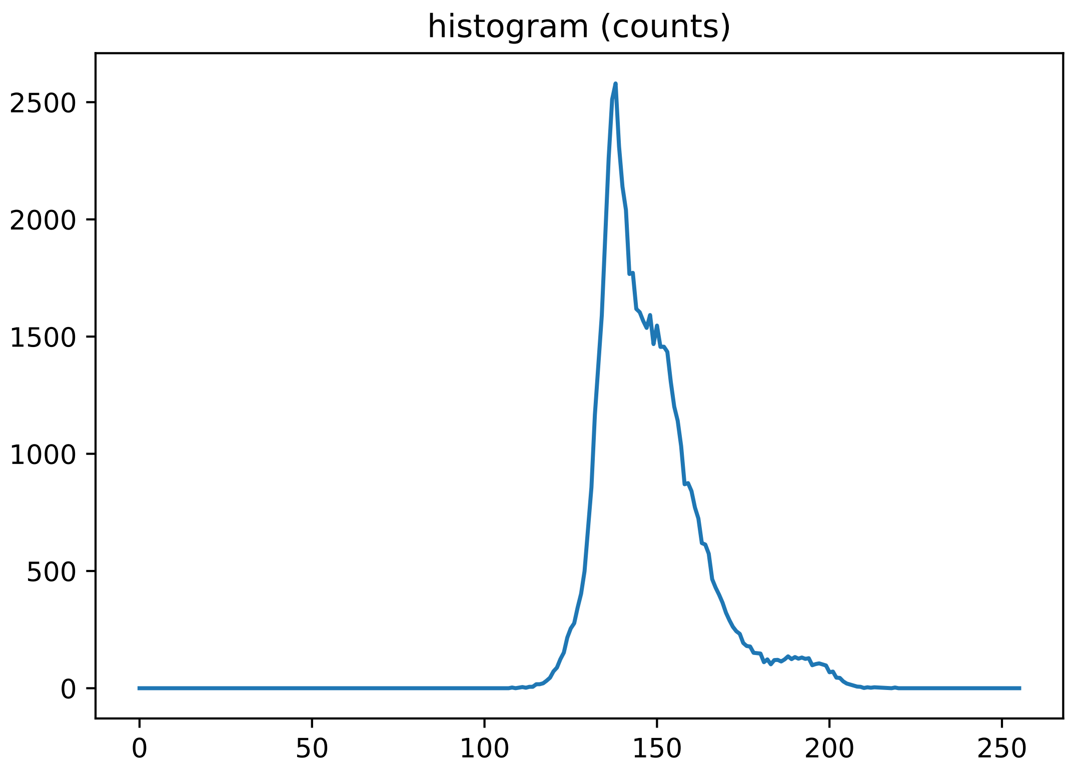


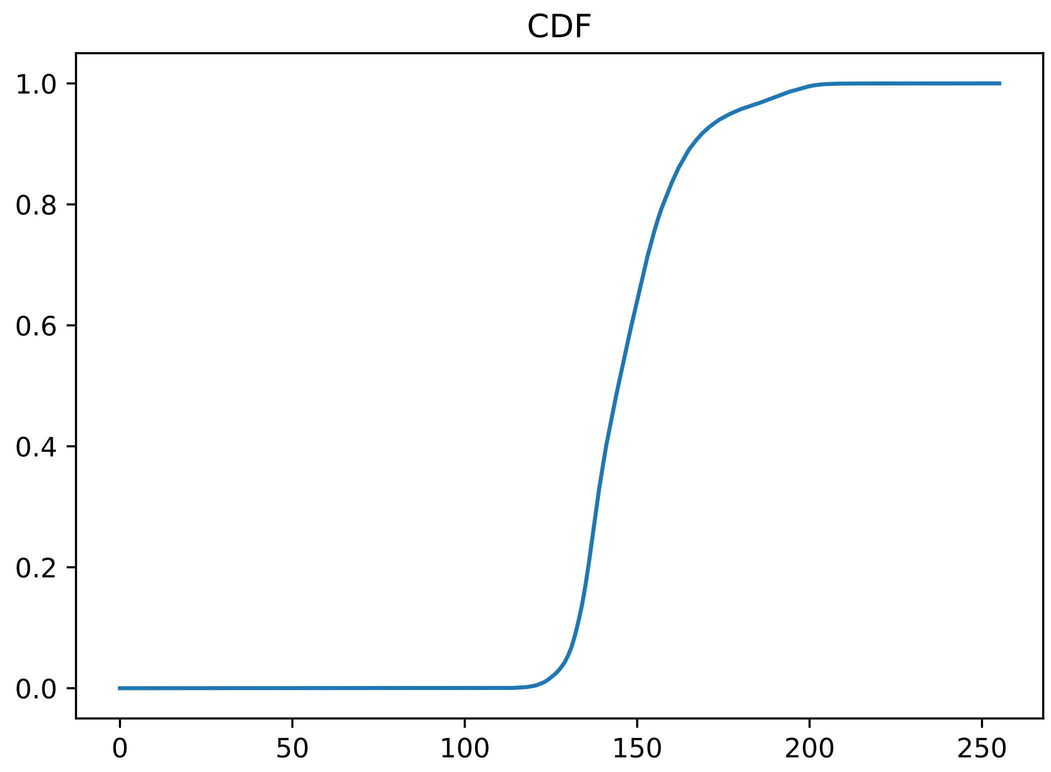
2.4 Histogram Equalization

- Check Wikipedia Histogram Equalization

```
## image size (gray scale):  (200, 300)
```

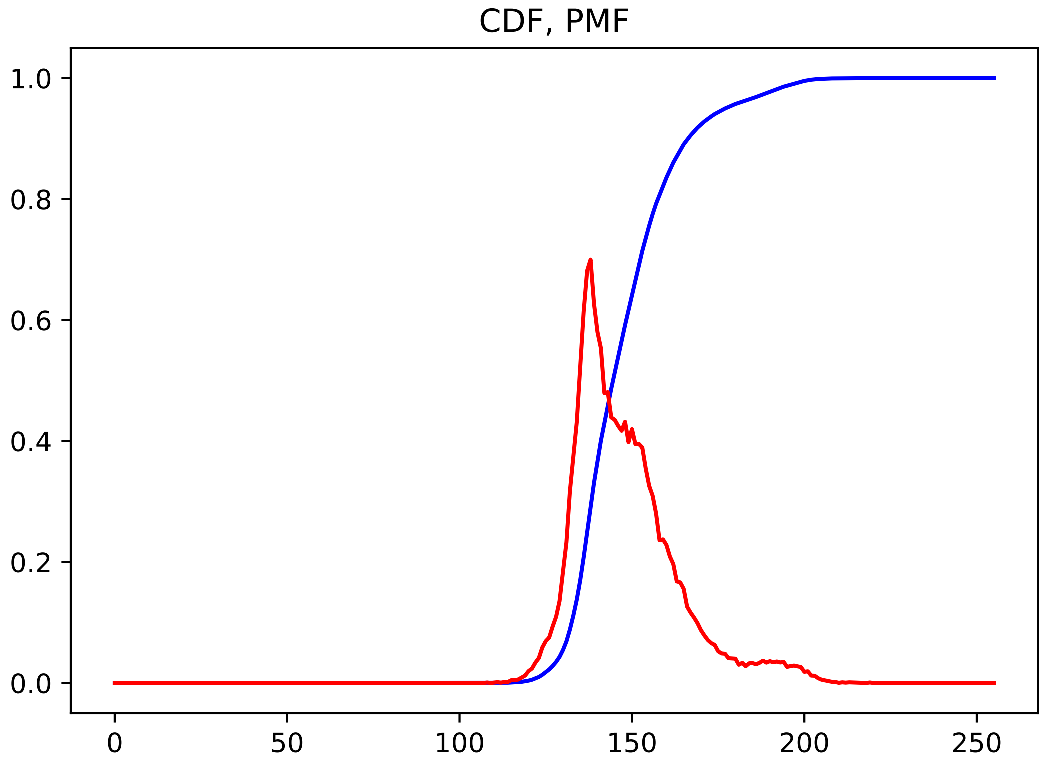
```
## cdf:  (256,)
```

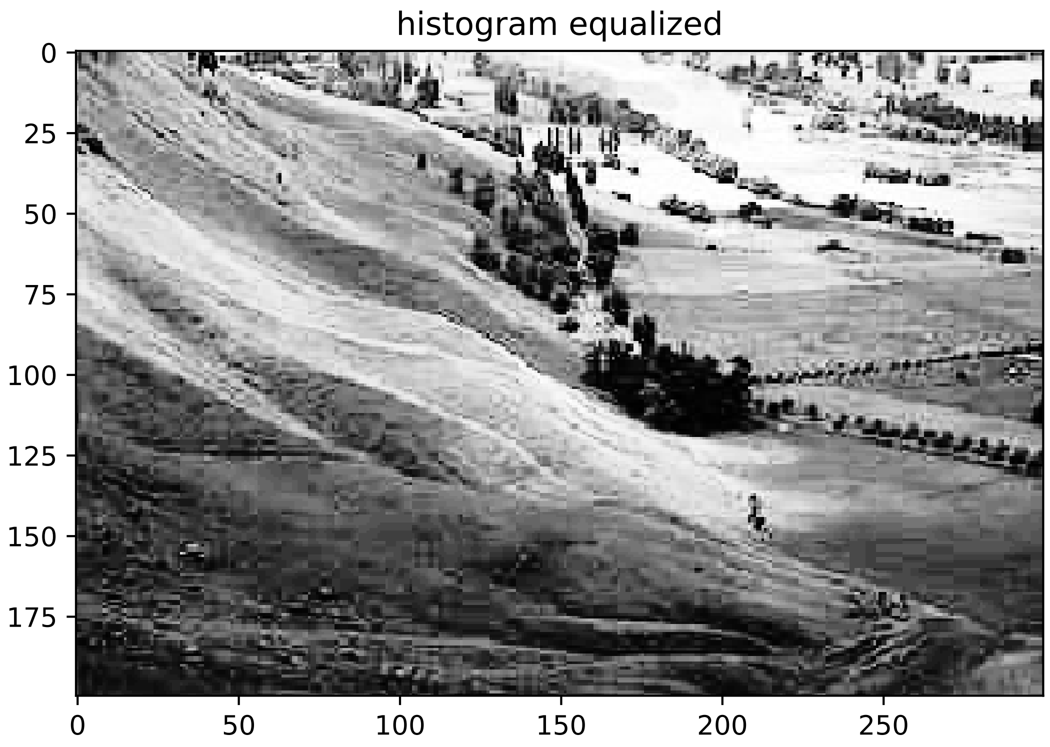

histogram (counts)
2500
2000
1500
1000
500
0
0
50
100
150
200
250



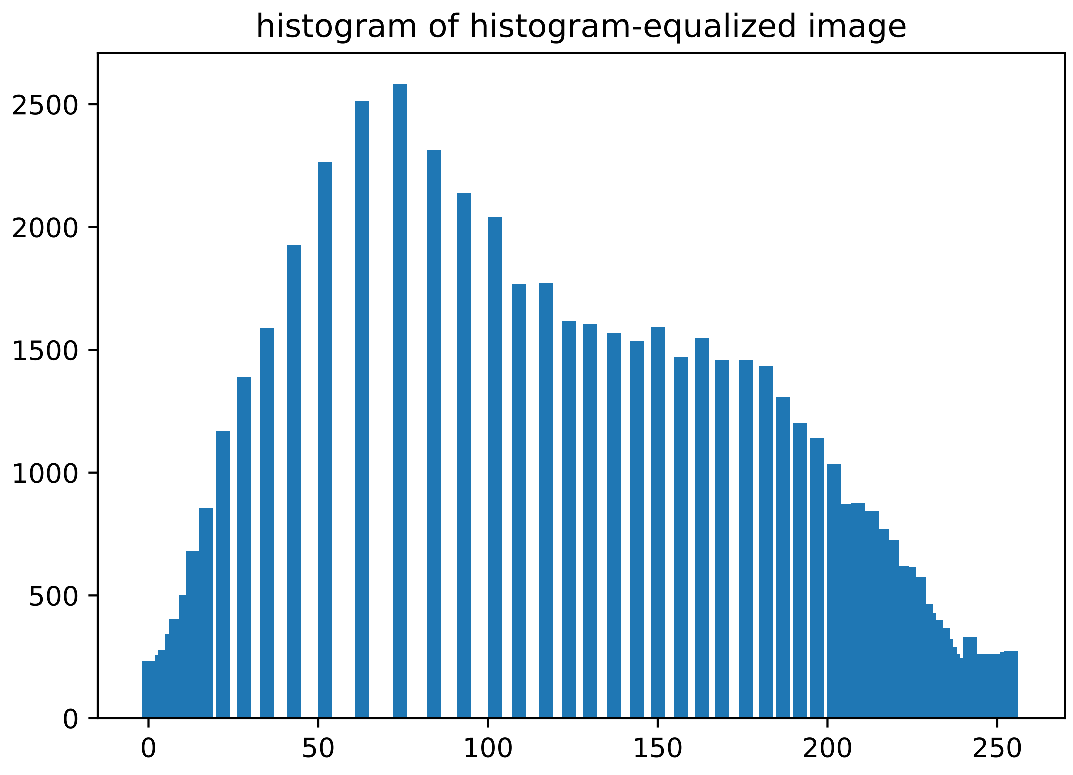
```
## max of pmf:  0.043
```

```
## [<matplotlib.lines.Line2D object at 0x7f11e20f2ef0>, <matplotlib.lines.Line2D object at 0x7f11e20cc0
```


CDF, PMF
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250


histogram equalized
0
25
50
75
100
125
150
175
0
50
100
150
200
250

```
## <BarContainer object of 256 artists>
```



```
# filename: histogram_equalization.py
# https://en.wikipedia.org/wiki/Histogram_equalization

import cv2
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

imagefilename = 'data/300px-Unequalized_Hawkes_Bay_NZ.jpg'
img = cv2.imread(imagefilename, cv2.IMREAD_GRAYSCALE)
print ('image size (gray scale): ', img.shape)

plt.imshow (img, cmap='gray')
plt.pause (1)
plt.close()

def histogram (ii):
    h = np.zeros(256)
    for val in ii.flatten():
        h[val] += 1
    return h
#

def make_cdf (pmf):
    cdf = np.zeros (pmf.shape)
    print ('cdf: ', cdf.shape)
    cdf[0] = pmf[0]
    for x in range(1, cdf.shape[0]):
        cdf[x] = cdf[x-1] + pmf[x]
    return cdf
#

# compute histogram
h = histogram (img)

# normalize to get PMF, Probability Mass Function
pmf = h / float(img.shape[0]*img.shape[1])

# Accumulate to get CDF, Cumulative Distribution Function
cdf = make_cdf (pmf)

plt.plot (h); plt.title ('histogram (counts)') # show the histogram
plt.pause (1) # seconds
plt.close()

plt.plot (cdf); plt.title('CDF') # check the CDF
plt.pause (1)
plt.close()

#
pmf_max = pmf.max()
print ('max of pmf: ', pmf_max)
plt.plot (range(cdf.shape[0]), cdf, 'b-', range(cdf.shape[0]), pmf*0.7/pmf_max, 'r-');
plt.title ('CDF, PMF')
plt.pause(1)
plt.close()

def histogram_equalization (img, cdf):
    ieq = np.zeros_like (img)
    for r in range(img.shape[0]):
        for c in range(img.shape[1]):
            pixelvalue = img[r,c]
            ieq[r,c] = np.clip(255. * cdf[pixelvalue], 0, 255).astype (np.uint8)
    #
    return ieq
#

# do it now
img_eq = histogram_equalization (img, cdf)

plt.imshow (img_eq, cmap='gray'); plt.title('histogram equalized')
plt.pause(1)
plt.close()

# check the histogram of the equalized image.
# verify what you did.

heq = histogram (img_eq)
```

```
plt.bar (range(0,256), heq, width=4)
plt.title('histogram of histogram-equalized image')
plt.pause (1)
plt.close()

# Now, make & plot the CDF of heq


# EOF
```

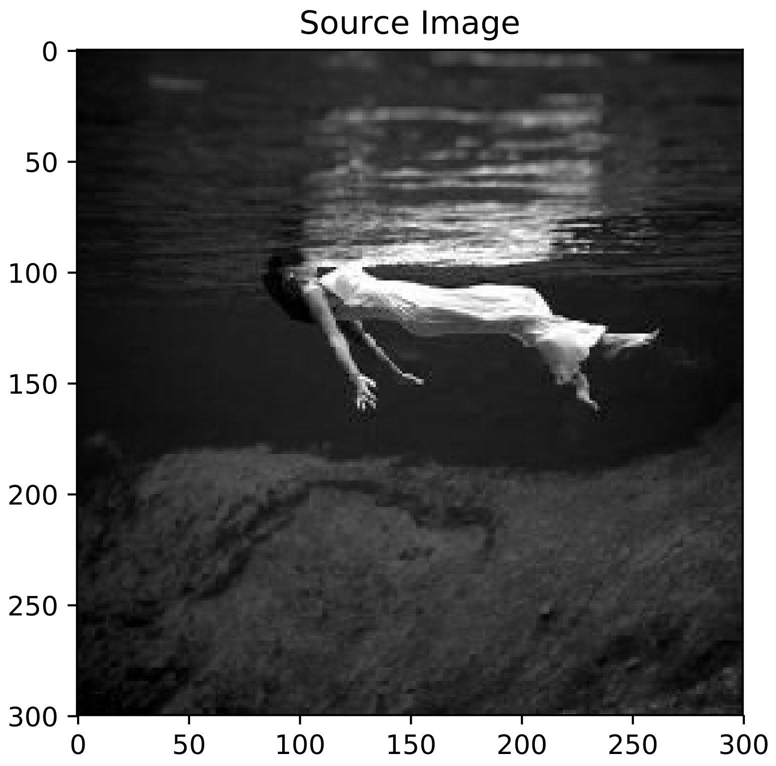
2.5 Gamma Correction

- Read Wikipedia Gamma Correction Page

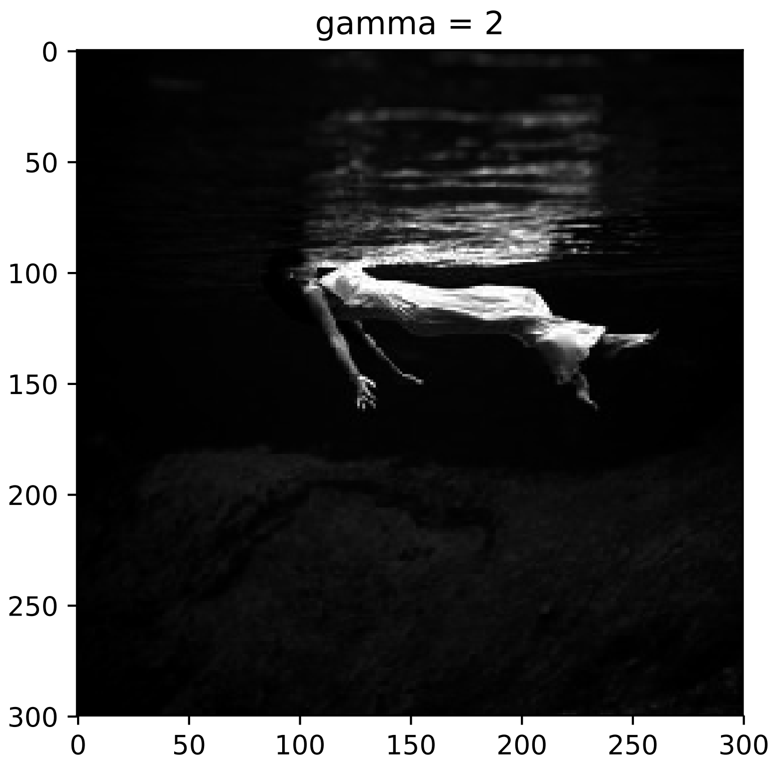
```
# filename gamma_correction.py
# https://en.wikipedia.org/wiki/Gamma_correction
# The image used from wikipedia seems to be from https://www.art.com/gallery/id--c23951/black-and-white

import sys
import numpy as np
import cv2
import matplotlib
matplotlib.use ('TkAgg')
import matplotlib.pyplot as plt
import imageio # this will be used to load an image file
import skimage # rgb <-> hsv conversion in [0,1] pixel scale

# show the image
def imshow (img, title=None):
    if img.ndim == 3:
        plt.imshow (img)
    else:
        plt.imshow (img, cmap='gray')

    if title is None: title = 'imshow'
    plt.title (title)
    plt.pause (1)
    plt.close ()
#

img = imageio.imread ('data/art.com.jpg') # it is an RGB format even though ...
if img is None:
    print ('image file open error')
    sys.exit ()
#
print (img.shape)
```

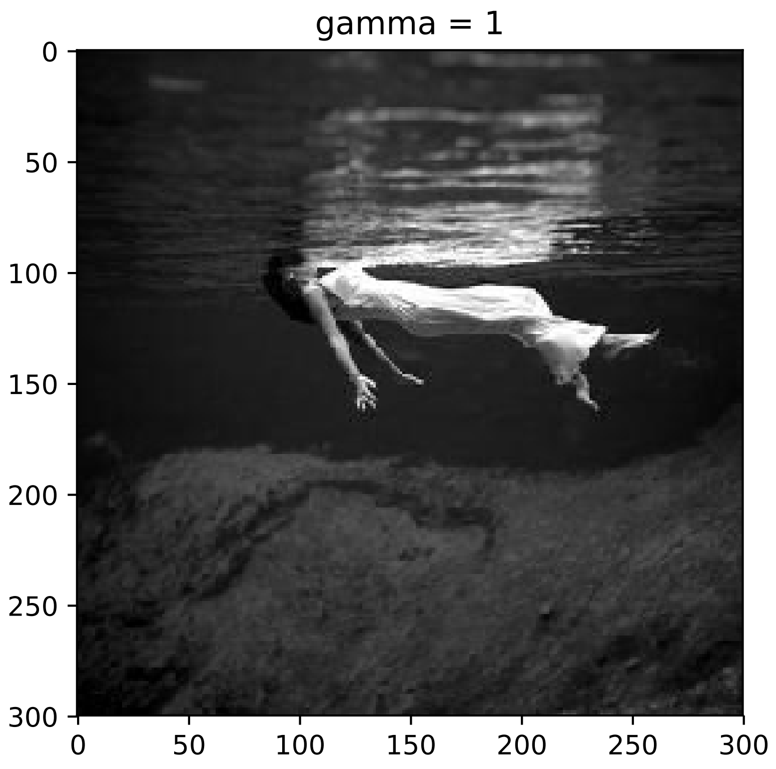
```
## (300, 300, 3)
```

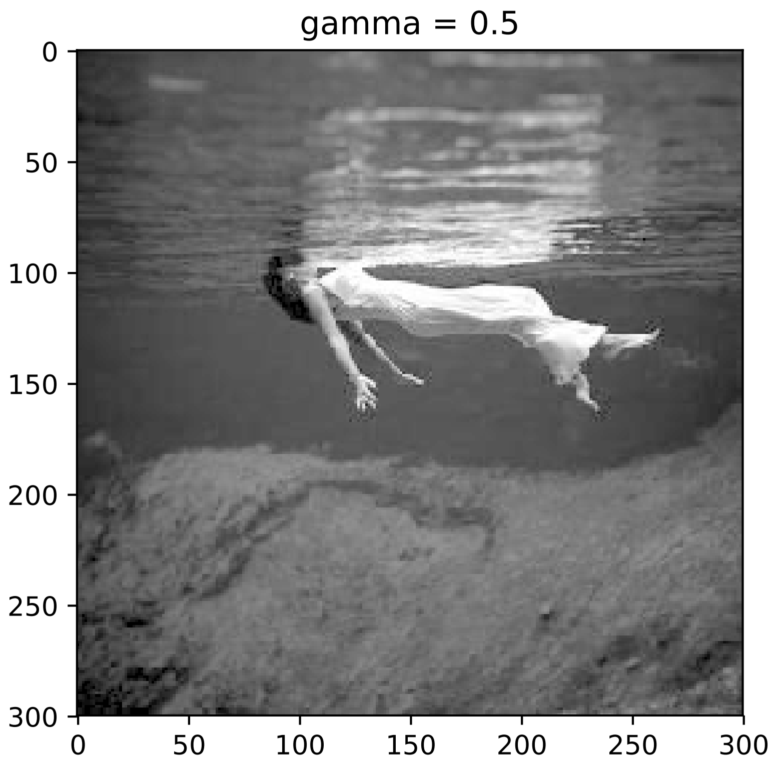
```
imshow (img, 'Source Image')
```

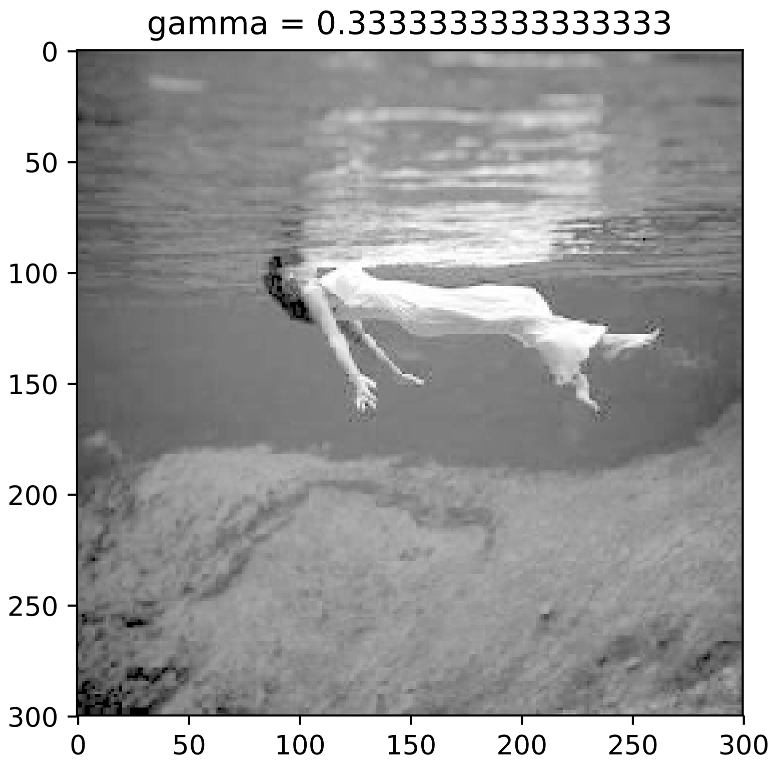


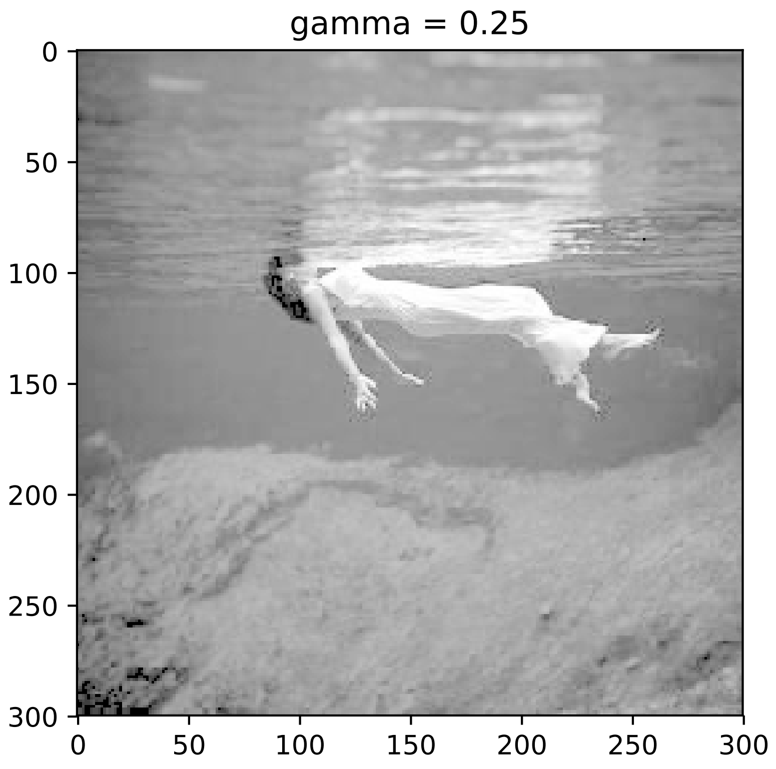
```
gammas = [2, 1, 1./2, 1./3, 1./4]

for gamma in gammas:
    ii = np.power(img/255., gamma)
    imshow (ii, 'gamma = {}'.format(gamma))
#

# Q. Plot histograms

# EOF
```


gamma = 2
0
50
100
150
200
250
300
0
50
100
150
200
250
300


gamma = 1
0
50
100
150
200
250
300
0
50
100
150
200
250
300


gamma = 0.5
0
50
100
150
200
250
300
0
50
100
150
200
250
300


gamma = 0.3333333333333333
0
50
100
150
200
250
300
0
50
100
150
200
250
300



2.6 HSV Color Space

HSV color representation is another popular way.

```
# filename: rgb_hsv.py
# REF: http://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_hsv.html

# RGB -> HSV is done with skimage.color.rgb2hsv()
#    * RGB, HSV data are assumed to be in the ranage [0,1]

import sys
import numpy as np
import cv2
import matplotlib
matplotlib.use ('TkAgg')
import matplotlib.pyplot as plt
import imageio # this will be used to load an image file
import skimage # rgb <-> hsv conversion in [0,1] pixel scale

# show the image
def imshow (img, title=None):
    if img.ndim == 3:
        plt.imshow (img)
    else:
```

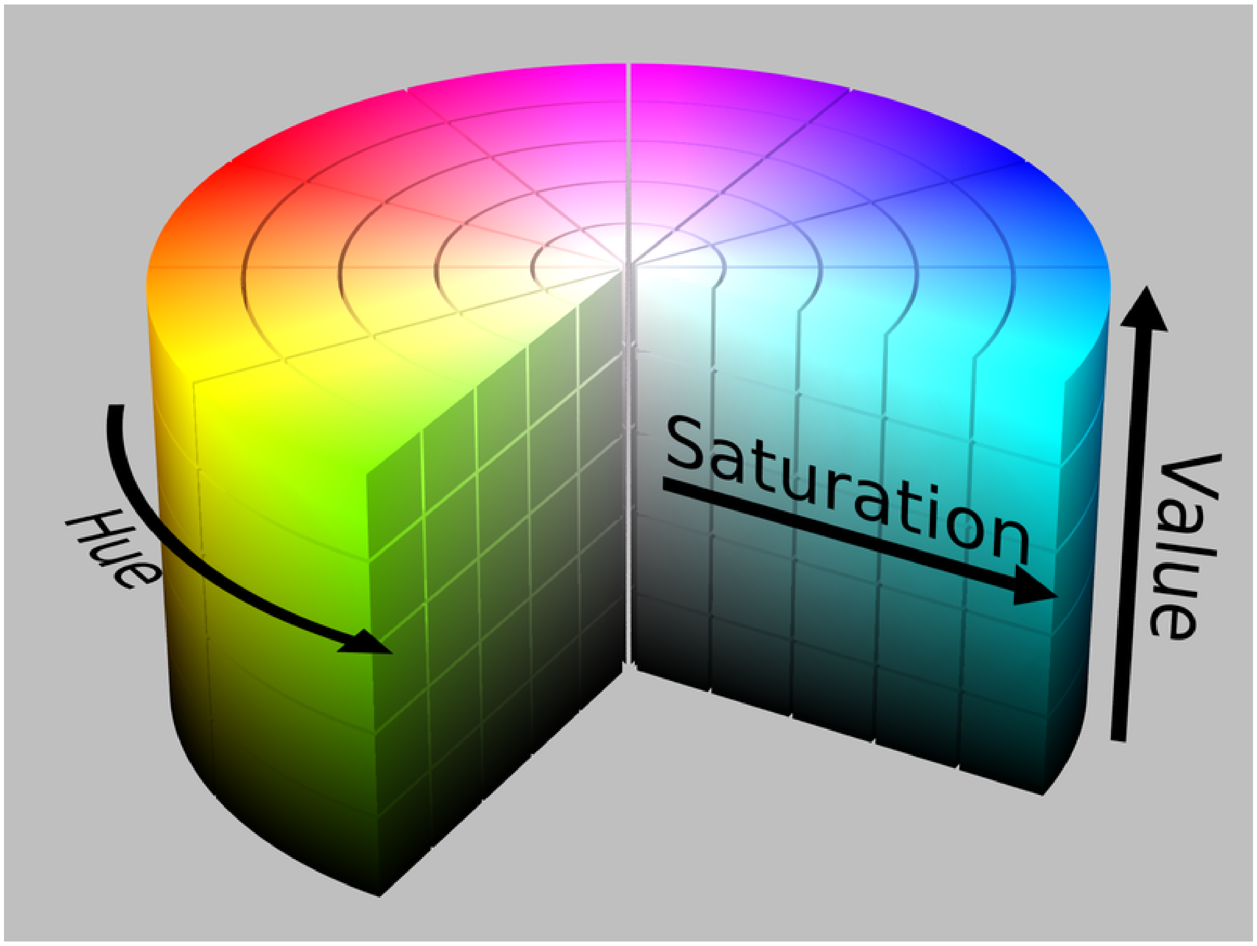
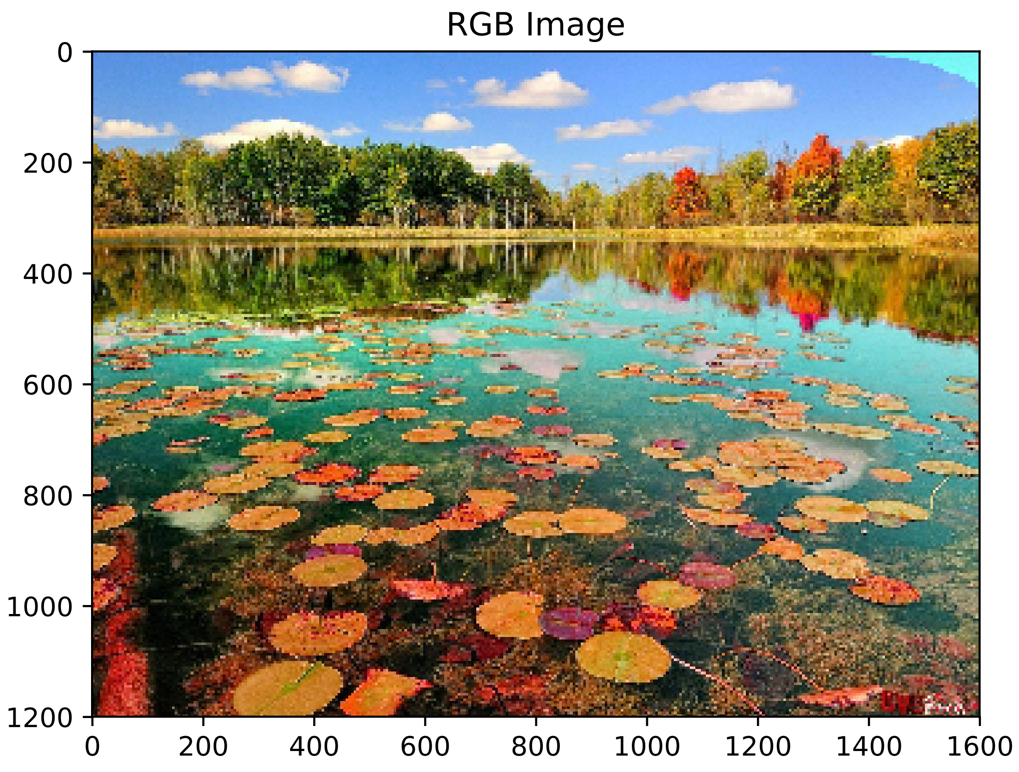
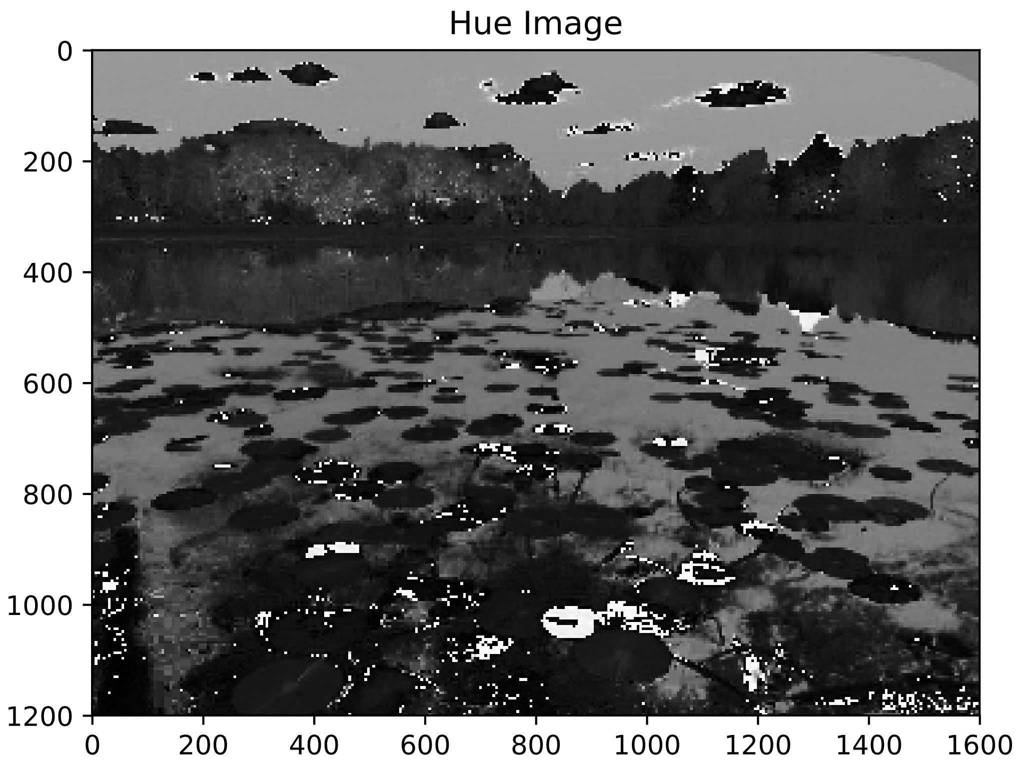


Figure 2.1: HSV Cylinder

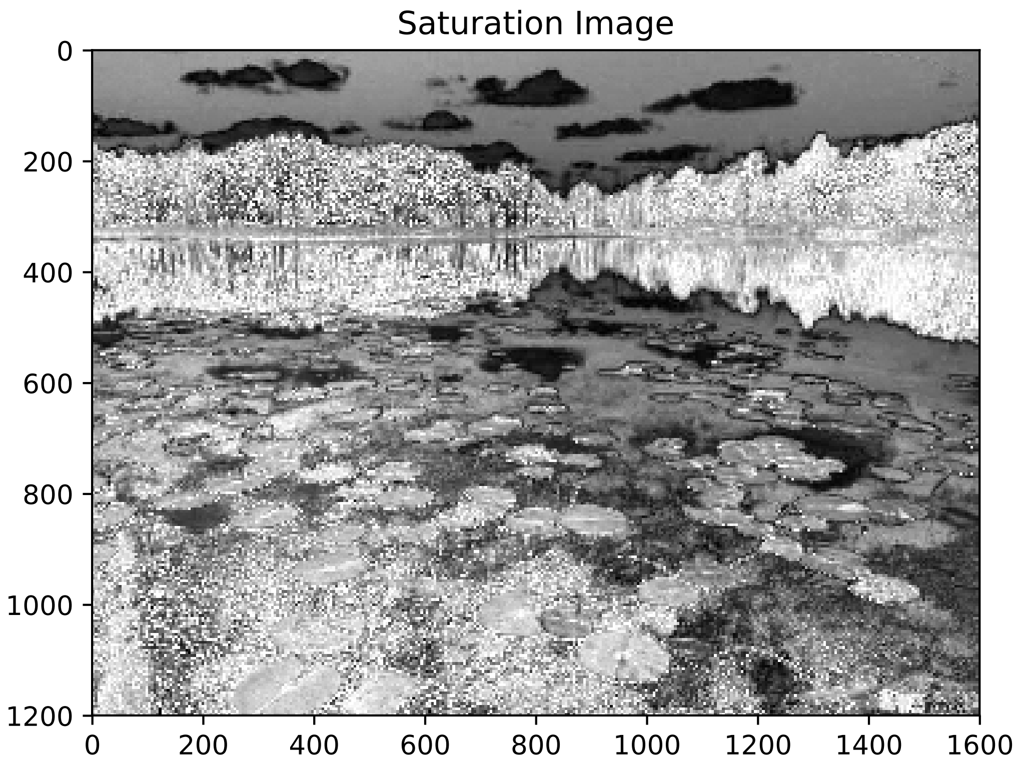
```
        plt.imshow (img, cmap='gray')

    if title is None: title = 'imshow'
    plt.title (title)
    plt.pause (1)
    plt.close ()
#

rgb = imageio.imread ('data/nature.jpg') #https://wallpapercave.com/pretty-nature-wallpapers
if rgb is None:
    print ('image file open error')
    sys.exit ()
#

imshow (rgb, 'RGB Image')

# The conversion assumes an input data range of [0, 1] for all color components.
```



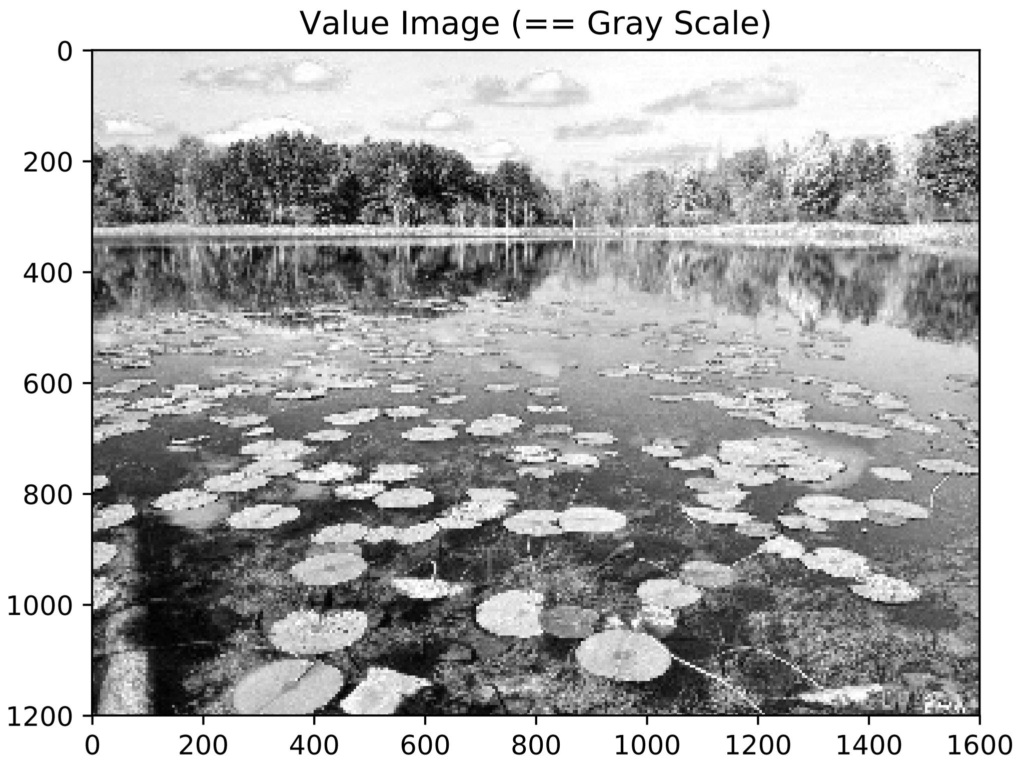
```
rgb01 = rgb/255.
hsv = skimage.color.rgb2hsv (rgb01)
hue = hsv[:,:,0]
saturation = hsv[:,:,1]
value = hsv[:,:,2]

imshow (hue, 'Hue Image')
```



```
imshow (saturation, 'Saturation Image')
```



```
imshow (value, 'Value Image (== Gray Scale)')
```



```
print ('HSV(Red)   = {}'.format(skimage.color.rgb2hsv([[[1.,0,0]]])))
```

```
## HSV(Red)   = [[[0. 1. 1.]]]
```

```
print ('HSV(Yellow)= {}'.format(skimage.color.rgb2hsv([[[1.,1.,0]]])))
```

```
## HSV(Yellow)= [[[0.16666667 1.         1.        ]]]
```

```
print ('HSV(Green) = {}'.format(skimage.color.rgb2hsv([[[0,1.,0]]])))
```

```
## HSV(Green) = [[[0.33333333 1.         1.        ]]]
```

```
print ('HSV(Blue)  = {}'.format(skimage.color.rgb2hsv([[[0,0,1.]]])))
```

```
## HSV(Blue)  = [[[0.66666667 1.         1.        ]]]
```

```
print ('HSV(White) = {}'.format(skimage.color.rgb2hsv([[[1.,1.,1.]]])))
```

```
## HSV(White) = [[[0. 0. 1.]]]
```

```
hsvpixel = [[[0.999, 1., 1.]]]
print ('RGB({}) = '.format(hsvpixel), skimage.color.hsv2rgb (np.array(hsvpixel)))
```

```
## RGB([[[0.999, 1.0, 1.0]]]) =  [[[1.    0.    0.006]]]
```

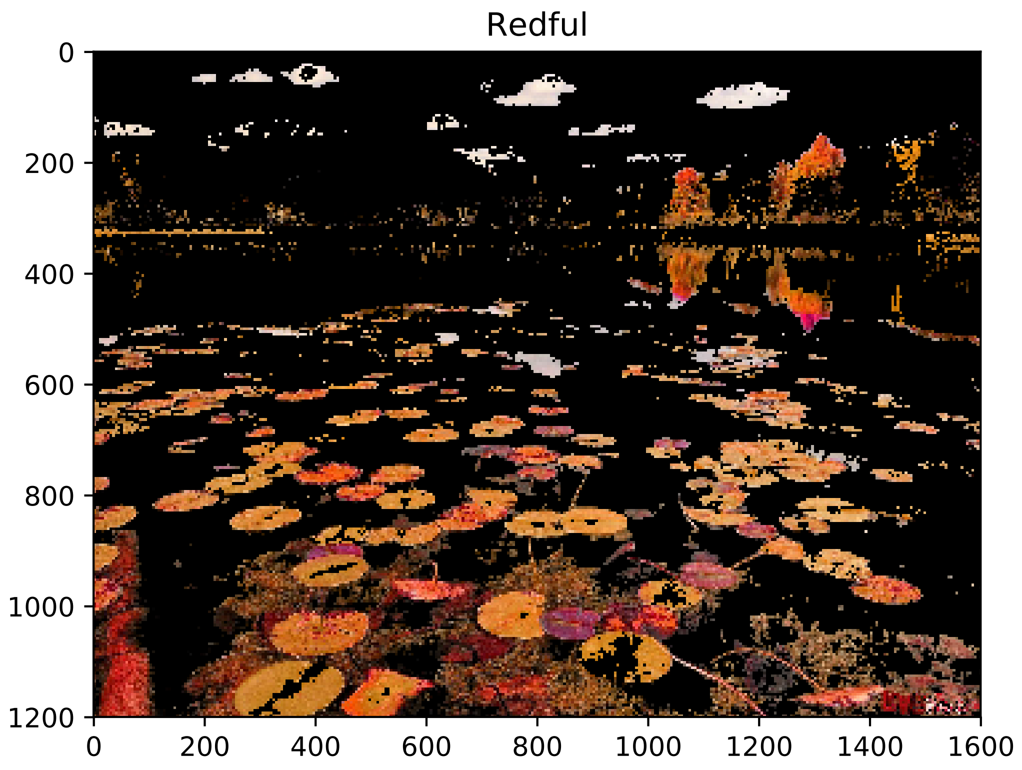
```
hsvpixel = [[[0., 1., 1.]]]
print ('RGB({}) = '.format(hsvpixel), skimage.color.hsv2rgb (np.array(hsvpixel)))
```

```
## RGB([[[0.0, 1.0, 1.0]]]) =  [[[1. 0. 0.]]]
```

```
print ('Now Extract Redful pixels only.')
```

```
## Now Extract Redful pixels only.
```

```
for r in range(hsv.shape[0]):
    for c in range (hsv.shape[1]):
        if (hsv[r,c,0] > 0.9 or hsv[r,c,0] < 0.1):
            pass # this is a red
        else:
            hsv[r,c,:] = [0,0,0] # black
#
rgb_redful = skimage.color.hsv2rgb (hsv)
imshow (rgb_redful, 'Redful')
```



```
imageio.imwrite ('data/redful.png', (rgb_redful*255).astype(np.uint8))

# Q. Can you remove white pixels in the clouds?
#    Hint: Examine the HSV Cylinder. ( hsv[r,c,1] > 0.5 )

# Q. Rotate the color along the HUE axis, make a video of the chage.
#    Step = 0, 0.01, 1

# EOF
```

Chapter 3

Non-Photorealistic Rendering

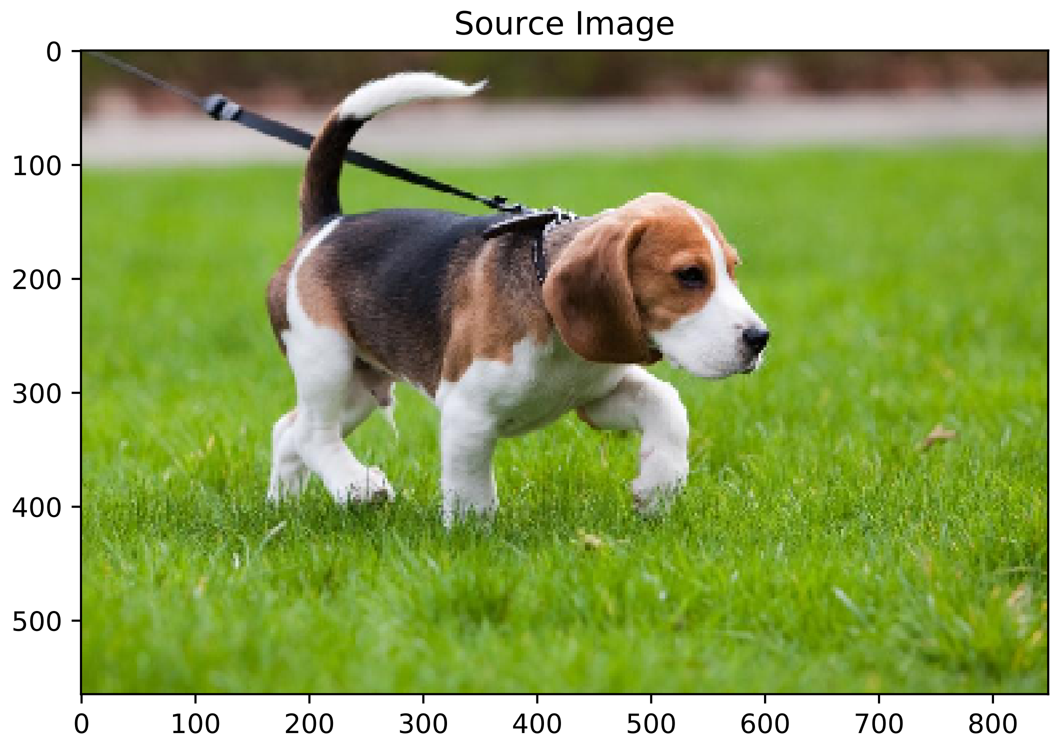
3.1 OpenCV non-photorealistic rendering

- Domain Transform for Edge-Aware Image & Video Processing

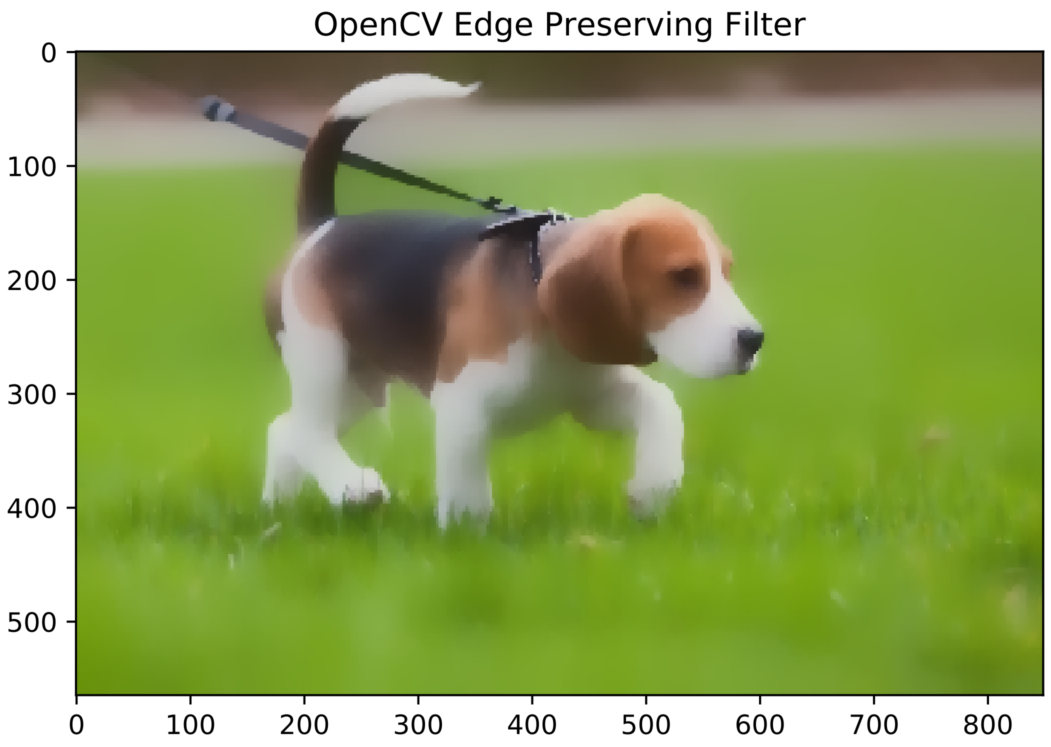
```
# filename: non-photorealistic.py

# https://www.learnopencv.com/non-photorealistic-rendering-using-opencv-python-c/
# https://docs.opencv.org/trunk/df/dac/group__photo__render.html

import sys
import numpy as np
import cv2
import matplotlib
matplotlib.use ('TkAgg')
import matplotlib.pyplot as plt
import imageio # this will be used to load an image file
import skimage # rgb <-> hsv conversion in [0,1] pixel scale

# show the image
def imshow (img, title=None):
    if img.ndim == 3:
        plt.imshow (img)
    else:
        plt.imshow (img, cmap='gray')

    if title is None: title = 'imshow'
    plt.title (title)
    plt.pause (1)
    plt.close ()
#

src = imageio.imread ('data/petinsider.com.jpg') # it is an RGB format even though ...
if src is None:
    print ('image file open error')
    sys.exit ()
#
print (src.shape)
```

```
## (565, 849, 3)
```

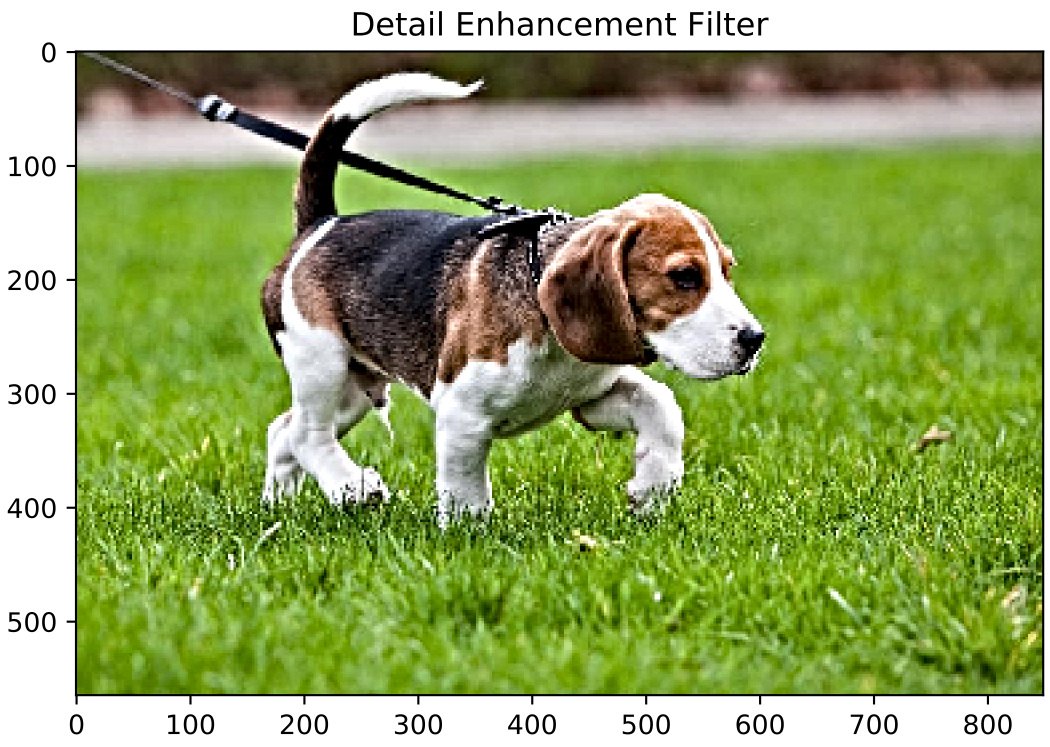
```
imshow (src, 'Source Image')
```



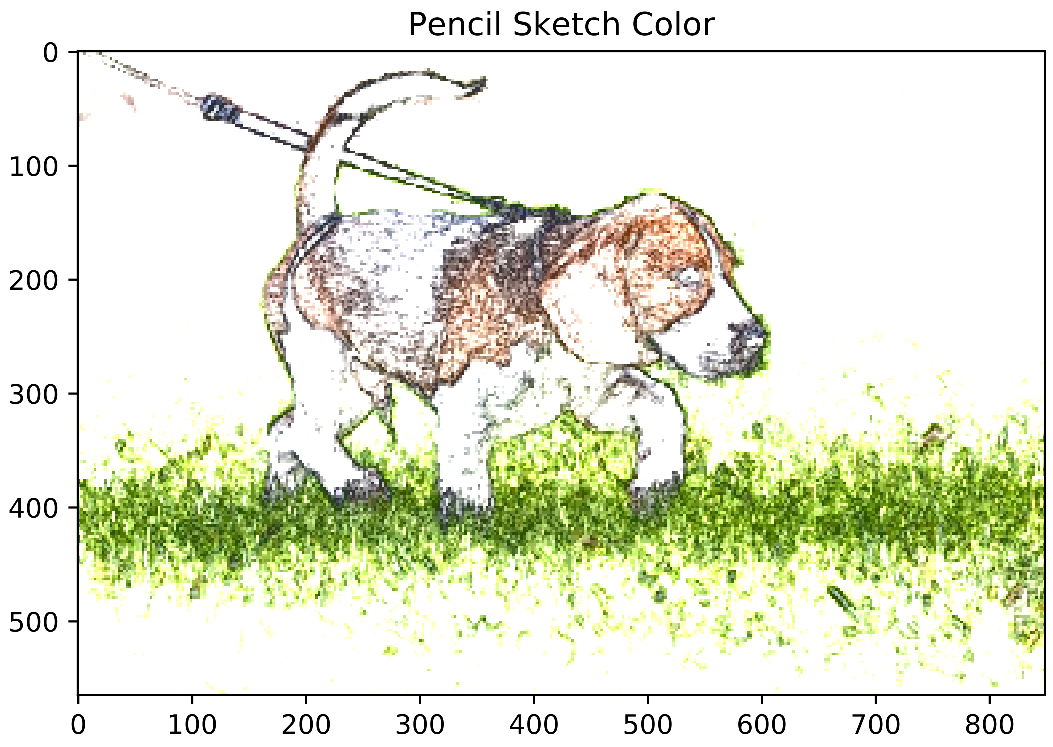
```
epf = cv2.edgePreservingFilter(src, flags=1, sigma_s=60, sigma_r=0.8)
imshow (epf, 'OpenCV Edge Preserving Filter')
```



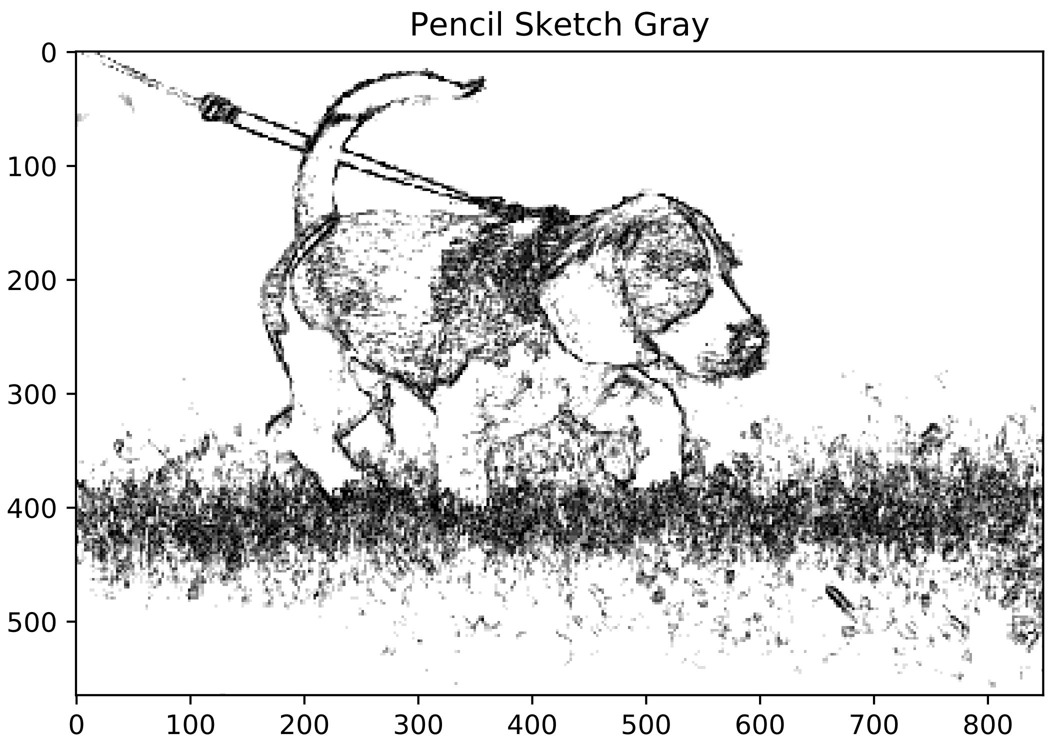
```
detf = cv2.detailEnhance (src, sigma_s = 10, sigma_r=0.2)
imshow (detf, 'Detail Enhancement Filter')
```



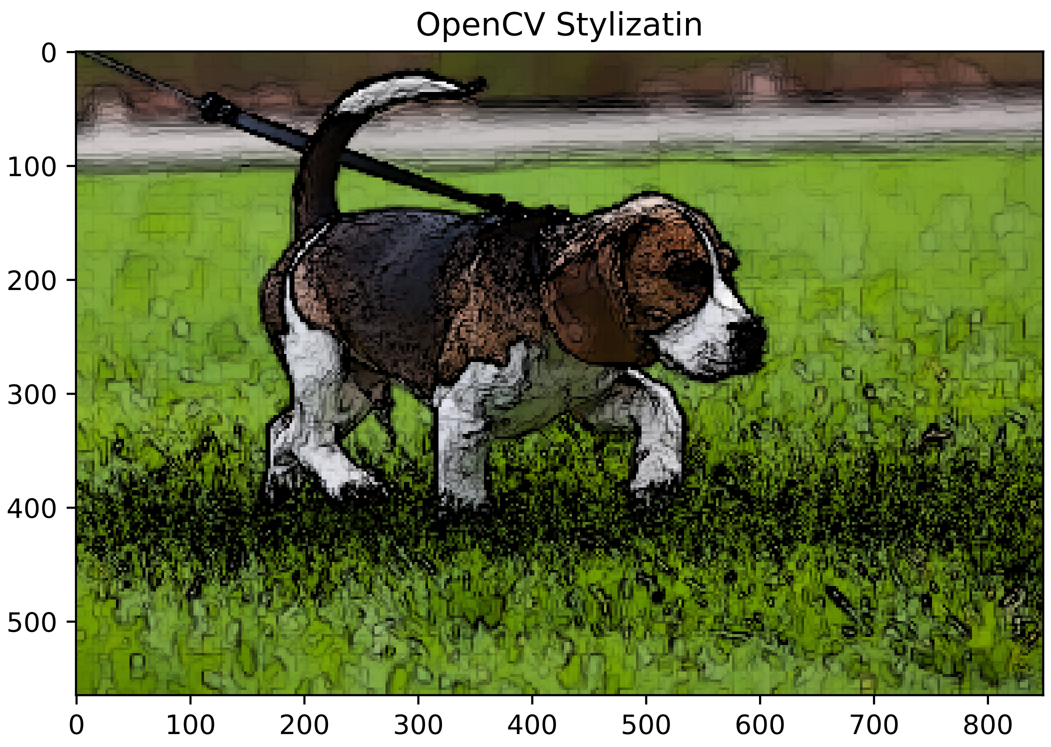
```
pencil_g, pencil_c = cv2.pencilSketch (src, sigma_s=60, sigma_r=0.07, shade_factor=0.1)
imshow (pencil_c, 'Pencil Sketch Color')
```



```
imshow (pencil_g, 'Pencil Sketch Gray')
```



```python
styl = cv2.stylization (src, sigma_r=0.05, sigma_s=50)
imshow (styl, 'OpenCV Stylizatin')

# EOF
```

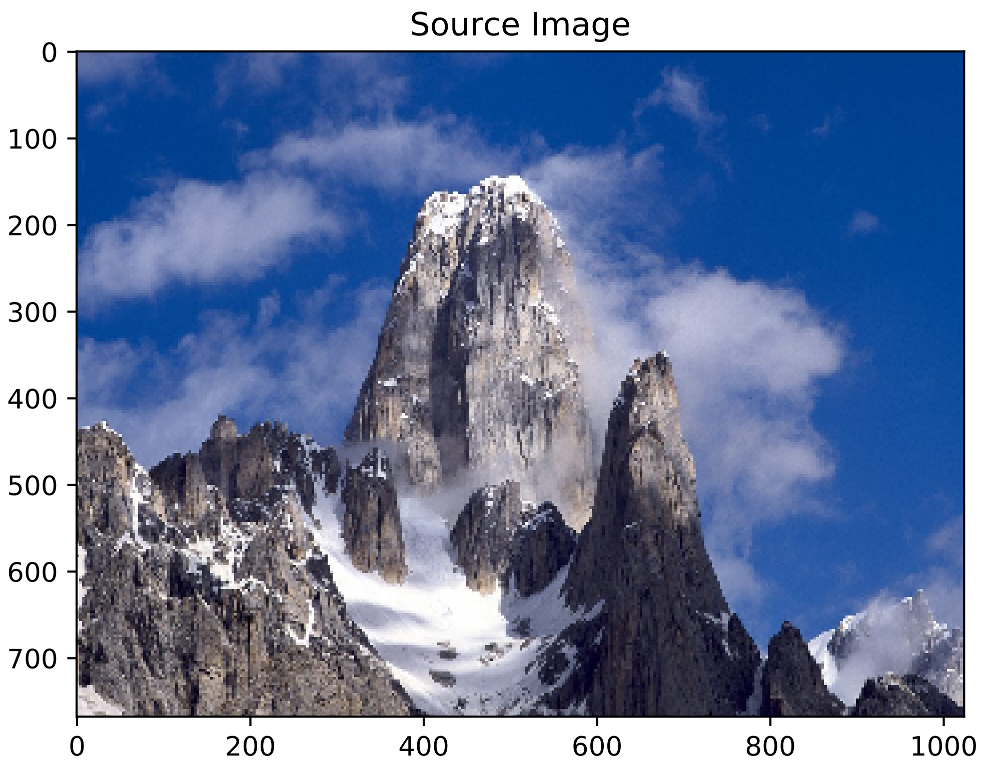


3.2 Hand-made cartoon-like filtering

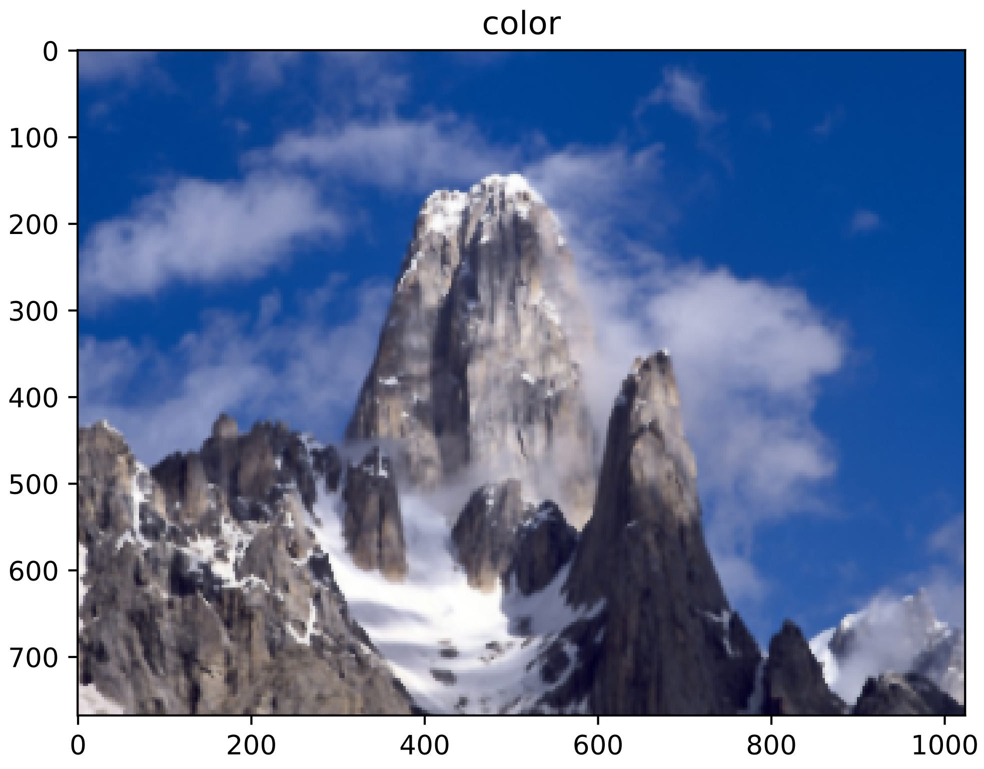
```
import cv2
import numpy as np
import imageio
import matplotlib.pyplot as plt

# show the image
def imshow (img, title=None):
    if img.ndim == 3:
        plt.imshow (img)
    else:
        plt.imshow (img, cmap='gray')

    if title is None: title = 'imshow'
    plt.title (title)
    plt.pause (1)
    plt.close ()
#

img = imageio.imread('data/karakoram-imgur.com.jpg')
if img is None:
    print ('image file open error')
    sys.exit ()
#
print (img.shape)
```

```
## (768, 1024, 3)
```

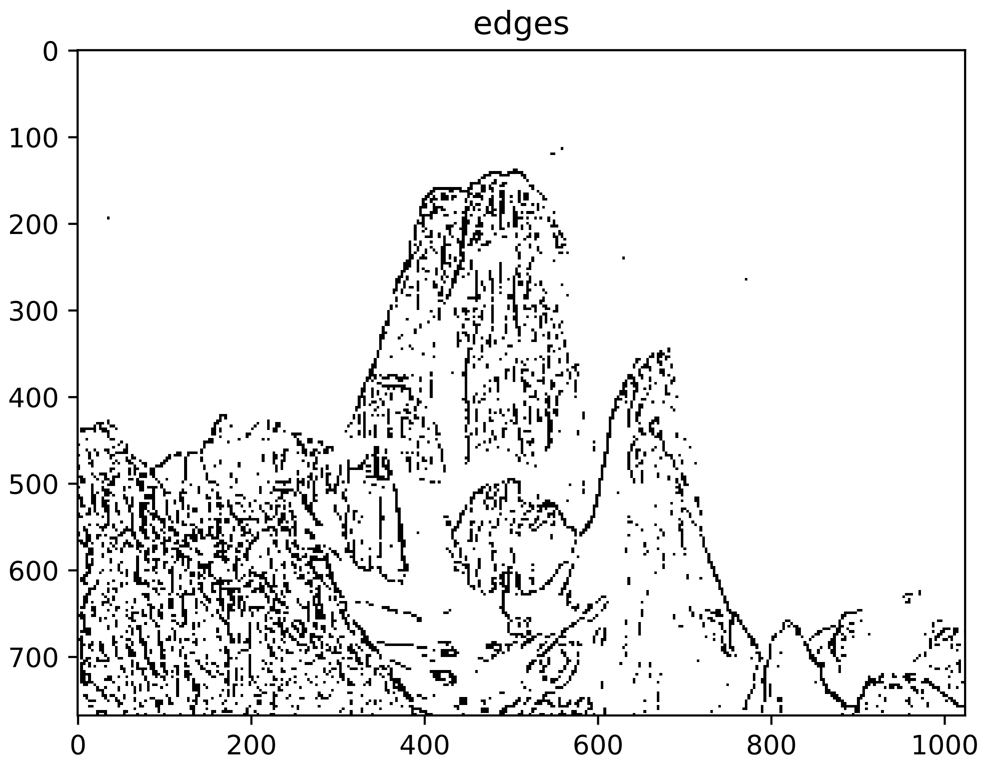
```
imshow (img, 'Source Image')


# 1) Edges
```



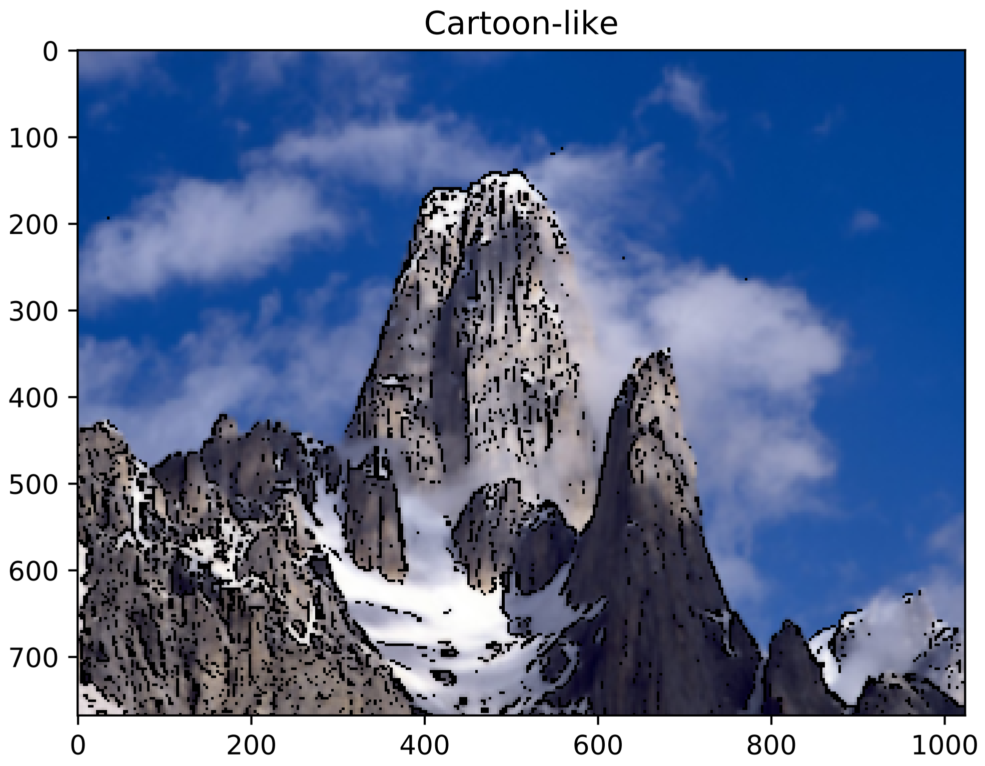
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = cv2.medianBlur(gray, 5)
edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 9)

# 2) Color
color = cv2.bilateralFilter(img, 9, 300, 300)

# 3) Cartoon
cartoon = cv2.bitwise_and(color, color, mask=edges)

# display
imshow(color, "color")
```



```python
imshow(edges, "edges")
```



```
imshow(cartoon, "Cartoon-like")

# EOF
```


Cartoon-like
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

Chapter 4

Video File Manipulation

4.1 Video File Read/Write

```
# filename: video-open.py
# https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_drawing_functions/py_d

import numpy as np
import cv2

video_file = 'data/avideo.mov'
cap = cv2.VideoCapture(video_file)

if cap.isOpened() is False:
    print ('video file open error: ', video_file)
#
width = cap.get (cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get (cv2.CAP_PROP_FRAME_HEIGHT)
nframes = cap.get (cv2.CAP_PROP_FRAME_COUNT)
fps = cap.get (cv2.CAP_PROP_FPS)
print (height, width, nframes, fps)

outvideofile = 'data/outvideo.mov'
out_wh = (640, 480)
outVideo = cv2.VideoWriter (outvideofile, cv2.VideoWriter_fourcc(*'XVID'), 30.0, out_wh)

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == False:
        break

    frame = cv2.resize(frame, dsize=out_wh)

    font = cv2.FONT_HERSHEY_SIMPLEX
    text_xy = (100, 200)
    frame = cv2.putText (frame, 'OpenCV', text_xy, font, 4, (0,255,255), 2, cv2.LINE_AA)

    cv2.imshow('frame', frame)
    outVideo.write (frame)

    if cv2.waitKey(30) == 27:
        break
#

cap.release()
outVideo.release()

cv2.destroyAllWindows()

# EOF
```

Notes:

1. The color image in opencv is BGR order, not RGB order.
2. `cap.get()` returns `float` numbers, not integer numbers.
3. The video `frame` obtained from `cap.read()` is a `numpy` array, in BGR order. If you want to know its color values at `(x,y)`, then try `frame[y,x]` and you will get the BGR at the location.
4. The fourcc `cv2.VideoWriter_fourcc()` is always confusing. Please search for a concrete explanation on it.
5. There is no way to deal with sound with `OpenCV`. Try another python module such as `moviepy`, see MoviePy for its documentation. Below is an example:

Try:

- Negative Film Effect Operation for a duration of the video
- Gray Scale
- Reversed-mode play

4.2 Video File Set Frame Position

This is a way of getting a video frame at a specifiic frame number.

- `videoCaputre().set(cv2.CAP_PROP_POS_FRAMES, nth_frame)`
- `videoCaputre().set(cv2.CAP_PROP_POS_AVI_RATIO, relative_position_0_to_1)`

```
# filename: video-lastframe.py

import sys
import numpy as np
import cv2

video_file = 'data/avideo.mov'
cap = cv2.VideoCapture(video_file)

if cap.isOpened() is False:
    print ('video file open error: ', video_file)
    sys.exit()
#
```

```
width = cap.get (cv2.CAP_PROP_FRAME_WIDTH)
height = cap.get (cv2.CAP_PROP_FRAME_HEIGHT)
nframes = cap.get (cv2.CAP_PROP_FRAME_COUNT)
fps = cap.get (cv2.CAP_PROP_FPS)
print (height, width, nframes, fps)

for i in range (int(nframes)):
    fcount = nframes - 1 - i
    cap.set (cv2.CAP_PROP_POS_FRAMES, int(nframes-1-i))
    ret, frame = cap.read()
    if frame is None:
        print ('frame None', i)
        continue
    if ret == False:
        print ('ret False', i)
        continue
    print ('frame read: ', i)

    frame = cv2.resize(frame, dsize=None, fx=.25, fy=.25)

    cv2.imshow('frame', frame)
    if cv2.waitKey(3) == 27:
        break
#

cap.release()
cv2.destroyAllWindows()
# EOF
```

- See: OpenCV VideoCapture Document for `cv2.CV_CAP_PROP_POS_FRAMES` which sets '0-based index of the frame to be decoded/captured next.'

4.3 MoviePy Example

If you need to manipulate audio & video together, `moviepy` is an option.

Check the site.

```
#
from moviepy.editor import *
import cv2
import numpy as np

videofile = 'data/avideo.mov'
video = VideoFileClip(videofile)
audio = video.audio
duration = video.duration # == audio.duration, presented in seconds, float
#note video.fps != audio.fps
print ('video.duration: ', video.duration, video.fps)
print ('audio.duration: ', audio.duration, audio.fps)

step = 0.1
```

```
for t in range(int(duration / step)): # runs through audio/video frames obtaining them by timestamp wit
    t = t * step
    if t > audio.duration or t > video.duration: break
    audio_frame = audio.get_frame(t) #numpy array representing mono/stereo values
    video_frame = video.get_frame(t) #numpy array representing RGB/gray frame

    cv2.imshow ('display', video_frame)
    if cv2.waitKey(25) == 27: break
#
```

Chapter 5

Methods

We describe our methods in this chapter.