개발자 가이드

모바일시스템공학과 : 32157629 한송이

32161620 박산희

소프트웨어학과 : 32157022 고수열

32157204 박상준

1) 소스 코드 입수 방법

github 주소 : https://github.com/gksthd1992/gajah

directory	설명
final files	최종 파이썬 코드는 final files directory에 위치한다. backend만 존재한다.
img	테스트 할 때 사용된 이미지
Web	웹 홈페이지와 py code를 합친 부분이다. final files 의 python code 기반이지만, html js php 와 같이 사용하게 되어 많은 수정이 들어갔다.
select_object	객체 추출에 대한 코드이다. html, js, py 코드가 들어있어 객체 추출 에 대한 front end , back end 부분을 알 수 있다.
image_edit	명도, 채도, RGB 추출에 대한 코드이다. python 코드이며, test image가 포함되어 있다.

2) 빌드 & 테스트 방법

Front end + back end

https://github.com/gksthd1992/gajah
Web 디렉토리에 html, pyhton, php, javascript 파일이 있다.

back end (python code only)

https://github.com/gksthd1992/gajah
의 final files 디렉토리 에 python 코드만 있다.

이 가이드에서는 final files 에 존재하는 python 코드에 대한 빌드방법을 작성하였다

1. image transfer

(1) python3가 돌아갈 환경을 만든다 (Jupyter Notebook에서 작업 시)

Jupyter Notebook을 다운 받아 환경을 준비한다.

(2) 이미지 변환에 쓰일 CNN모델인 vgg-19 model을 다운받아 작업 환경 내에 준비시킨다.

(다운 :http://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-19.mat)

(3) 파이썬 내에 필요한 라이브러리들을 import한다.

```
import os
import cv2
import sys
import numpy as np
import scipy.io
import scipy.misc
import tensorflow as tf
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from PIL import Image
%matplotlib inline
```

(4) 변환시킬 사진(content_image)과 원하는 스타일의 사진(style_image)의 이름을 입력하고 사진의 WIDTH와 HEIGHT를 입력하여 사진을 준비한다.

```
# Output folder for the images.
OUTPUT_DIR = 'output11/'
# Style image to use.
STYLE_IMAGE = 'Ss.jpg'
# Content image to use.
CONTENT_IMAGE = 'OPENCV (1).jpg'
# Image dimensions constants.
IMAGE_WIDTH = 300
IMAGE_HEIGHT = 400
COLOR_CHANNELS = 3
```

(5) 부분합성을 위해 img에 contet_image를 준비시킨다.

(*original= 완성된 이미지의 이름으로 원하는 이름으로 수정 가능하다.)

```
[ ]   #오픈CV
      img = CONTENT_IMAGE
      original = "art.jpg"
```

(6) 부분합성에 필요한 def combine_two(input1,input2) 함수와 def wartershed(input_im) 함수를 준비시킨다.

```
def combine_two(input1, input2):

    img1 = cv2.imread(input1)
    img2 = cv2.imread(input2)
def wartershed(input_im):

    img = cv2.imread(input_im)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

※combine_two함수는 두 개의 이미지를 합치는 함수
※wartershed함수는 사진 내에 object의 Edge를 찾아내어 배경색을 변경하는 함수

*combine_two(input1,input2)함수 내에 dsize로 최종사진의 원하는 넓이와 높이로 조정한다.

```
img1 = cv2.resize(img1, dsize=(300, 400))
img2 = cv2.resize(img2, dsize=(300, 400))
```

(7) 이미지 합성과정에서 contents_image와 style_image중 어느 이미지에 가까운 결과를 낼지를 결정하는 값들을 설정해주고 다운받은 VGG_MODEL을 VGG_MODEL변수에 넣는다.

```
NOISE_RATIO = 0.6
BETA = 5
ALPHA = 100
VGG_MODEL = 'imagenet-vgg-verydeep-19.mat'
MEAN_VALUES = np.array([123.68, 116.779, 103.939]).reshape((1,1,1,3))
```

*위에 α/β값이 작아질수록 content보다는 style에 더 치중된 결과

(8) load_vgg_model을 이용하여 VGG-MODEL을 준비시킨다.
(load_vgg_mode함수 내에는 VGG-MODEL을 준비시키는

def _weights, def_relu, def_conv2d, def_conv2d_relu, def_avgpool 함수가 들어간다.)

```
def load_vgg_model(path):
    """
```

(9) style이미지의 손실 값을 계산하는 함수를 준비시킨다.

(내부에 def _gram_matrix, def _style_loss 함수 포함)

```
def style_loss_func(sess, model):
    """
```

```
def generate_noise_image(content_image, noise_ratio = NOISE_RATIO):
    """
    Returns a noise image intermixed with the content image at a certain ratio.
    """
    noise_image = np.random.uniform(
            -20, 20,
            (1, IMAGE_HEIGHT, IMAGE_WIDTH, COLOR_CHANNELS)).astype('float32')
    # White noise image from the content representation. Take a weighted average
    # of the values
    input_image = noise_image * noise_ratio + content_image * (1 - noise_ratio)
    return input_image

def load_image(path):
    image = scipy.misc.imread(path)
    # Resize the image for convnet input, there is no change but just
    # add an extra dimension.
    image = np.reshape(image, ((1,) + image.shape))
    # Input to the VGG model expects the mean to be subtracted.
    image = image - MEAN_VALUES
    return image

def save_image(path, image):
    # Output should add back the mean.
    image = image + MEAN_VALUES
    # Get rid of the first useless dimension, what remains is the image.
    image = image[0]
    image = np.clip(image, 0, 255).astype('uint8')
    scipy.misc.imsave(path, image)
```

(10) noise_image를 생산하는 함수, 이미지 준비 함수, 이미지 저장 함수를 준비시킨다.

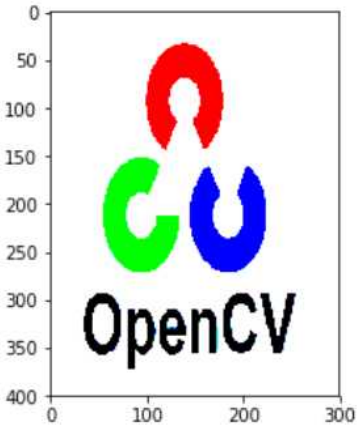
(11) sess변수에 tensorflow세션을 열어 준비한다.

```
[ ]  sess = tf.InteractiveSession()
```

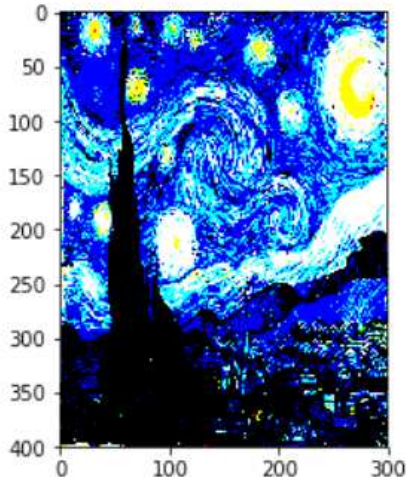
(12) contenst이미지를 앞서 준비한 load_image함수를 통해 준비시킨 후 결과를 확인한다.

```
[ ]  content_image = load_image(CONTENT_IMAGE)
     imshow(content_image[0])
```

(13) style이미지를 앞서 준비한 load_image함수를 통해 준비시킨 후 결과를 확인한다.




```
[ ]   style_image = load_image(STYLE_IMAGE)
      imshow(style_image[0])
```



(14) 앞서 준비한 load_vgg_model함수를 이용하여 model을 준비한다.

```
[ ]   model = load_vgg_model(VGG_MODEL)
```

(15) 앞서 준비한 generate_noise_image함수를 이용하여 noise_image을 준비한다.

```
      input_image = generate_noise_image(content_image)
      imshow(input_image[0])
```

(16) 앞서 준비한 텐져플로우 세션을 작동한다.

```
[ ]   optimizer = tf.train.AdamOptimizer(2.0)
      train_step = optimizer.minimize(total_loss)
```

```
[ ]   sess.run(tf.initialize_all_variables())
      sess.run(model['input'].assign(input_image))
```

(17) 반복횟수를 사용자 정의로 정한다.

```
sess.run(tf.initialize_all_variables())
```

```
sess.run(model['input'].assign(content_image))
content_loss = content_loss_func(sess, model)
```

```
# Construct style_loss using style_image.
sess.run(model['input'].assign(style_image))
style_loss = style_loss_func(sess, model)
```

```
# Instantiate equation 7 of the paper.
total_loss = BETA * content_loss + ALPHA * style_loss
```

```
ITERATIONS = 1000
```

(18) 반복횟수대로 작동하는 반복코드를 실행한다.

```
sess.run(tf.initialize_all_variables())
sess.run(model['input'].assign(input_image))
for it in range(ITERATIONS):
    sess.run(train_step)
    if it%100 == 0:
        # Print every 100 iteration.
        mixed_image = sess.run(model['input'])
        print('Iteration %d' % (it))
        print('sum : ', sess.run(tf.reduce_sum(mixed_image)))
        print('cost: ', sess.run(total_loss))

        if not os.path.exists(OUTPUT_DIR):
            os.mkdir(OUTPUT_DIR)

        filename = 'output11/%d.png' % (it)
        save_image(filename, mixed_image)
```

*코드를 실행 시 반복횟수대로 잘 반복이 되는지 실행창을 통해 확인한다.

```
Iteration 0
sum :  15079098.0
cost:  150604480000.0
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:28: DeprecationWarning: `imsave` is deprecated!
`imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imwrite`` instead.
Iteration 100
sum :  13737216.0
cost:  3533488000.0
Iteration 200
sum :  12930030.0
cost:  1784777200.0
Iteration 300
```

(19) 완성된 이미지를 art.jpg로 저장한다.

(사용자 지정 원하는 이름으로 수정 가능 But, 5번과정에 original의 이름과 맞춰줘야 한다.)

```
save_image('art.jpg', mixed_image)
```

(20) 부분합성을 원할 시 combine_two함수를 이용하여 위에서 완성시킨 original함수와 wartershed()함수를 매개변수로 결합한다.

```
combine_two(wartershed(img),original)
```

2. select object

(1) python3가 돌아갈 환경을 만든다 (Jupyter Notebook에서 작업 시)
Jupyter Notebook을 다운 받아 환경을 준비한다.

(2) 객체를 추출할 이미지를 준비하고 코드가 존재하는 폴더에 넣는다.

(3) 필요한 모듈을 import 한다.

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

(4) 이미지를 불러오고 저장할 경로를 지정한다.

```
#저장할 이미지 경로
OUTPUT_DIR = 'input/output11/'

#불러올 이미지 경로
input_im = "input/opencv_ori.jpg"

# 각 객체를 저장하는 이미지 이름 list, 객체가 많으면 추가해야 한다.
name = ["input/ret0.jpg",
        "input/ret1.jpg",
        "input/ret2.jpg",
        "input/ret3.jpg",
        "input/ret4.jpg",
        "input/ret5.jpg",
        "input/ret6.jpg",
        "input/ret7.jpg"]
```

(5) get_total_ret 함수
객체의 개수를 반환하는 함수이다.

```
def get_total_ret(input_im):
    #input_im 으로 이미지를 불러올 경로를 저장한다.
    img = cv2.imread(input_im)
    img = cv2.resize(img, dsize=(300, 400))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

(6) opencv, 모폴로지 연산등을 사용하여 객체를 구분하고 그 객체의 수를 반환한다.

```
kernel = np.ones((3, 3), np.uint8)
"""
morphologyEx 연산을 사용하여 작은 객체를 없애준다. noise 제거
interation 이 증가할수록 반복 횟수가 증가하여 많은 노이즈가 사라지지만, 객체도 작아진다.
"""
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

"""
noise를 제거하고 남은 객체를 더 확실하게 키워준다.
interation 이 증가할수록 반복 횟수가 증가한다.
"""
sure_bg = cv2.dilate(opening, kernel, iterations=3)

dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist_transform, 0.5 * dist_transform.max(), 255, 0)
sure_fg = np.uint8(sure_fg)

unknown = cv2.subtract(sure_bg, sure_fg)
#나눌 수 있는 객체의 갯수를 반환한다.
ret, markers = cv2.connectedComponents(sure_fg)
return ret
```

(7) watershed를 사용하는 함수

이미지를 객체 단위로 분리하여 jpg 형태로 저장한다.
이미지 사이즈를 300,400으로 조정한다.

```
def obj_wartershed(input_im,want):
    img = cv2.imread(input_im)
    img = cv2.resize(img, dsize=(300, 400))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

(8) get total ret과 같은 기능을 한다.

```
kernel = np.ones((3, 3), np.uint8)
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)

sure_bg = cv2.dilate(opening, kernel, iterations=3)

dist_transform = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
ret, sure_fg = cv2.threshold(dist_transform, 0.5 * dist_transform.max(), 255, 0)
sure_fg = np.uint8(sure_fg)

unknown = cv2.subtract(sure_bg, sure_fg)

ret, markers = cv2.connectedComponents(sure_fg)
```

(9) 원하는 객체를 입력하면 그 부분 외에 검정으로 처리해 준다.

```
markers = markers + 1
markers[unknown == 255] = 0
markers = cv2.watershed(img, markers)
img[markers == -1] = [0, 0, 0]

want = input("원하는 번호를 입력하세요: ")

ret = int(ret)
for i in range(ret + 1):
    if i != int(want):
        img[markers == i] = [0, 0, 0]

cv2.imwrite(name[want], img)
return img
```

(10) make_obj_img 함수는 원하는 객체까지 모든 객체의 이미지를 저장해준다.

```
def make_obj_img(input, ret) :
    for i in range(ret):
        obj_wartershed(input,i+1)
```

(4) 함수 호출

전체 객체를 각각의 이미지로 저장한다.

```
total_ret = get_total_ret(input_im)
make_obj_img(input_im,total_ret)
```

3. get_rgb

이미지의 평균 RGB 값, 가장 많이 사용되는 RGB 값,
많이 사용되는 RGB를 비율로 나타낸 bar 그래프를 반환한다.

(1) python3가 돌아갈 환경을 만든다 (Jupyter Notebook에서 작업 시)
Jupyter Notebook을 다운 받아 환경을 준비한다.

(2) 변환할 이미지를 준비하고 코드가 존재하는 폴더에 넣는다.

(3) 필요한 module을 import 한다.

```
import os
import cv2
import sys
import numpy as np
import scipy.io
import scipy.misc
import tensorflow as tf
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from PIL import Image
from sklearn.cluster import KMeans
```

```
%matplotlib inline
```

(4) 구글 드라이브에서 colaboratory 로 돌린다면 drive를 mount 한다.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import os
os.chdir("/content/drive/My Drive/image transfer")
!ls
```

(5) 이미지를 불러온다.

```
image = cv2.imread("rabbit2.jpg")
```

(6) BGR 형태의 이미지를 RGB 형태의 이미지로 바꾼다.

```
# 채널을 BGR -> RGB로 변경
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

image = image.reshape((image.shape[0] * image.shape[1], 3)) # height, width 통합
print(image.shape)
# (25024, 3)
```

(7) k 값에 RGB 추출의 개수를 지정한다.

```
k = 5 # 5개로 나눈다
clt = KMeans(n_clusters = k)
clt.fit(image)
```

(8) k의 개수만큼 많이 사용되는 RGB를 추출한다.

```
def centroid_histogram(clt):
    # grab the number of different clusters and create a histogram
    # based on the number of pixels assigned to each cluster
    numLabels = np.arange(0, len(np.unique(clt.labels_)) + 1)
    (hist, _) = np.histogram(clt.labels_, bins=numLabels)

    # normalize the histogram, such that it sums to one
    hist = hist.astype("float")
    hist /= hist.sum()

    # return the histogram
    return hist
```

(9) 추출한 RGB를 bar graph 형태의 이미지로 저장한다.
바 그래프를 출력한다.

```
def plot_colors(hist, centroids):
    # initialize the bar chart representing the relative frequency
    # of each of the colors
    bar = np.zeros((50, 300, 3), dtype="uint8")
    startX = 0

    # loop over the percentage of each cluster and the color of
    # each cluster
    for (percent, color) in zip(hist, centroids):
        # plot the relative percentage of each cluster
        endX = startX + (percent * 300)
        cv2.rectangle(bar, (int(startX), 0), (int(endX), 50),
                      color.astype("uint8").tolist(), -1)
        startX = endX

    # return the bar chart
    return bar


bar = plot_colors(hist, clt.cluster_centers_)
```

(10) 가장 많은 비율의 RGB를 수치로 반환한다.

```
def largest_rate_rgb(hist,total,rgb):
    biggest=-1
    num=-1
    for i in range(total) :
        if(biggest<hist[i]) :
            biggest=hist[i]
            num=i
    return rgb[num]
big = largest_rate_rgb(hist,k,clt.cluster_centers_)
print(big)
```

(11) 평균 RGB를 하나의 수치로 반환한다.

```
def average_rgb(hist, total, rgb):
    average = [0,0,0]
    sum = [0,0,0]
    for i in range(total) :
        sum = (sum + hist[i]*rgb[i]*100)
    average = sum/100
    return average
```

4. change_color

이미지의 명도와 채도를 –100에서 100 의 수치에 대해서 변환한다.

(1) python3가 돌아갈 환경을 만든다 (Jupyter Notebook에서 작업 시)
Jupyter Notebook을 다운 받아 환경을 준비한다.

(2) 객체를 추출할 이미지를 준비하고 코드가 존재하는 폴더에 넣는다.

(3) 필요한 모듈을 import한다.

```
import os
import cv2
import sys
import numpy as np
import scipy.io
import scipy.misc
import tensorflow as tf
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from PIL import Image
import matplotlib.image as mpimg
from sklearn.cluster import KMeans

%matplotlib inline
```

(4) 저장된 경로에서 필요한 이미지를 불러온다.

```
# 이미지 파일을 읽어온다
img = mpimg.imread("rabbit2.jpg", cv2.IMREAD_COLOR)
```

(5) 이미지의 정보를 저장한다.

```
imgplot = plt.imshow(img)
plt.show()

height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]
```

(6) BGR의 형식에서 HSV 형식으로 변환한다.

명도, 채도는 HSV 형식이기 때문이다.

```
# BGR to HSV 변환
img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
print(img_hsv[399][1])
imgplot = plt.imshow(img_hsv)
#print(img_hsv[img_hsv.__len__()-1]) # last array
plt.show()

#img_result = cv2.bitwise_and(img, img, img_hsv)
#imgplot = plt.imshow(img_result)
#plt.show()
middle = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)
imgplot = plt.imshow(middle)
```

(7) 채도 변환 수치를 입력받는다.

```
enter_sat = int(input("채도를 입력하세요 (-100 ~ 100): "))
print(enter_sat)
```

(8) 채도를 변환하는 함수.

이미지의 정보와 수치를 input으로 받고 화면에 출력한다.

```
def change_saturation(width,height,img_hsv,enter_sat):
    for x in range(width-1) :
        for i in range(height-1) :
            if(enter_sat < 0) :
                img_change[i][x][1] = img_hsv[i][x][1] - img_hsv[i][x][1] * (abs(enter_sat)/100)
            if(enter_sat >= 0) :
                img_change[i][x][1] = img_hsv[i][x][1] + img_hsv[i][x][1] * (enter_sat/100)
            #img_change[i][x][1] = img_hsv[i][x][1] + img_hsv[i][x][1] * enter_sat
            img_change[i][x][0] = img_hsv[i][x][0]
            img_change[i][x][2] = img_hsv[i][x][2]


    imgplot = plt.imshow(img_change)
    plt.show()

    img_result = cv2.cvtColor(img_change, cv2.COLOR_HSV2BGR)
    imgplot = plt.imshow(img_result)
    plt.show()
```

(9) 명도 수치를 입력한다.

```
enter_value = int(input("명도를 입력하세요 (-100 ~ 100): "))
print(enter_value)
```

(10) 명도를 변환하는 함수.

이미지의 정보와 수치를 input으로 받고 화면에 출력한다.

```
def change_value(width,height,img_hsv,enter_value):
    for x in range(width-1) :
        for i in range(height-1) :
            if(enter_value < 0) :
                img_change[i][x][0] = img_hsv[i][x][0] - img_hsv[i][x][0] * (abs(enter_value)/100)
            if(enter_value >= 0) :
                img_change[i][x][0] = img_hsv[i][x][0] + img_hsv[i][x][0] * (enter_value/100)
            #img_change[i][x][1] = img_hsv[i][x][1] + img_hsv[i][x][1] * enter_sat
            img_change[i][x][2] = img_hsv[i][x][2]
            img_change[i][x][1] = img_hsv[i][x][1]


    imgplot = plt.imshow(img_change)
    plt.show()

    img_result = cv2.cvtColor(img_change, cv2.COLOR_HSV2BGR)
    imgplot = plt.imshow(img_result)
    plt.show()

    cv2.imwrite('Final_value.jpg', img_result)
```

3) 데일리 빌드 및 테스트 환경 설정 방법

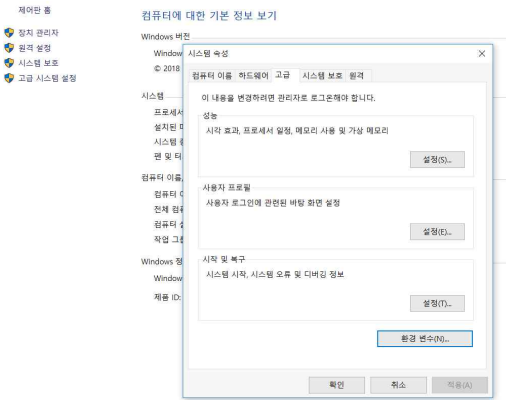
openCV를 위한 환경 구축하기

1. 아나콘다 설치하기

https://hobbang143.blog.me/221461726444

다운로드 후 설치 (관리자 권한)

환경 변수 설정하기
: 내 컴퓨터 (우클릭)-> 속성 -> 고급 시스템 설정 -> 고급 -> 환경 변수 클릭



시스템 변수 -> Path 선택, 편집 클릭 -> (없는 경우)새로 만들기 / (있는 경우) 경로 고치기
C:₩Anaconda3₩Scripts (경로는 설치경로에 따라 다를 수 있다) 추가하기

anaconda prompt를 관리자 권한으로 실행한다.

2. anaconda prompt를 관리자 권한으로 실행한다.

https://blog.naver.com/wjs7347/221497712851

가상환경 환경을 만들어준다
> conda create --name TestEnv python=3
(TestEnv 는 가상환경 이름)

확인한다.
> conda env list

가상환경 활성화를 한다.
> conda activate 가상환경이름

```
(base) C:₩WINDOWS₩system32>conda env list
# conda environments:
#
base                  *  C:₩ProgramData₩Anaconda3
SanheeEnv                C:₩Users₩32161620₩AppData₩Local₩conda₩conda₩envs₩SanheeEnv


(base) C:₩WINDOWS₩system32>conda activate SanheeEnv
```

openCV 설치하기

conda list 해서 라이브러리 확인하기

open cv가 없으면 설치한다
>pip install (필요 라이브러리 이름)OpenCV-Python

conda list 해서 라이브러리가 설치되었는지 확인하기

파이참에 가상환경 적용하기

파이참 실행 -> File -> New Project
existing interpreter에서 가상환경 저장 경로 불러오기(python.exe)

프로젝트 폴더 우클릭 -> New -> Python File 생성.

필요한 library 설치 및 제거 (pip로 설치하였으면 pip로만 삭제하여야 한다.)

conda list 해서 라이브러리 확인하기

>pip install (필요 라이브러리 이름)OpenCV-Python

>pip uninstall (불필요 라이브러리 이름)OpenCV-Python

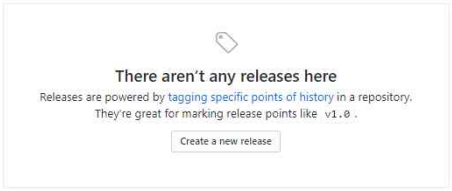
필요한 library를 설치를 했는데도 import 오류가 난다면 library를 삭제 후 다시 설치하는 것을 권한다.
아나콘다 prompt 창에서 관리자 권한으로 실행한다.

pycharm 말고 jupyter notebook 실행할 때
anaconda > jupyter notebook

4) 릴리즈 방법

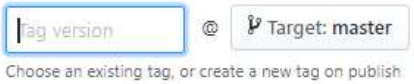
git hub에서 작업을 진행한다.

repository로 이동하여 release 상단 버튼을 클릭한다.

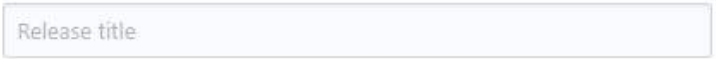


create a new release를 클릭한다.

branch에 tag를 기반으로 release 하게 되어있으므로 release 할 branch와 tag를 입력한다.



title을 작성한다.



설명을 작성한다.

완성본이 아니라면 온전하지 않음을 사용자에게 알린다.

release 한다.

5) 버그 확인 및 처리 방법

- 버그 리포팅으로 받은 경우

버그 리포팅으로 받은 메일을 바탕으로 어디서, 어떤 형태의 에러인지 확인을 한다.
웹에서 에러가 났을 경우 다른 컴퓨터 or OS 버전으로 접속을 하여 작동을 시켜본다. 버그 리포팅을 보내준 사람과 같은 에러가 있을경우, 에러 부분을 상세하게 책과 사이트를 통해 찾아본 후 수정한다.
다른 에러일 경우, 크게 3가지로 나눌 수 있다. 코드와 코드의 데이터 송수신 문제, 서버의 문제, 코드의 문제가 있다.

첫 번째의 경우, html-py, html-js , js-php, php-py 등의 코드와 코드 사이의 연결 문제가 있을 수 있다.
이는 ajax, hyperlink, action 시의 data 이동 등을 주로 고려하여 코드를 수정한다.

두 번째의 경우, 서버의 문제인지를 확인한다.
cmd 창에서 요구되는 module이 설치가 되어있는지 확인한다.
서버가 올바르게 작동하는지 사용한 웹서버를 확인한다.
방화벽이 올바르게 설정되어있는지 확인한다. 아파치에서는 port번호 80을 제공하므로, 80에 해당하는 ip들의 접근은 해제해야 한다.

세 번째의 경우, py,php,js 자체의 코드 오류인지를 확인한다.
python에서 image의 경로가 올바른지를 확인한다. 또한, 저장 경로도 확인한다.
module이 설치가 되어있는지 확인한다. module이 적합한 버전인지 확인한다. 낮은 버전이라면 update를 진행해야 하지만, update를 하는 과정에서 필요한 메소드가 더 이상 없을 수도 있으니 메소드 확인을 하면서 버전을 조절한다.
마지막으로, 문법이 정확한지 본다. 코드를 드래그를 하면서 코드가 뒤엉키거나 사라질 수 있다.

아래는 버그리포팅과는 상관없이 추가로 자주 발생하는 에러에 대한 상세한 내용이다.

- 객체의 분할이 잘되지 않을 때

여러 종류의 사진(어두운 사진, 객체가 여러 개인 사진, 밝은 사진, 흐린 사진, 객체의 구분이 명확하지 않은 사진) 등을 input으로 넣어서 test를 해본다.
git 프로젝트 사이트의 testing branch에서 check_object code를 돌려서 어떤 부분에서 문제가 있는지를 확인한다.
check_object : watershed 알고리즘 사용하여 객체 추출을 할 때, 각 단계별로 잘 실행되는지 단계별 이미지 출력을 보여준다.

- import 에러

anaconda에서 pip install <library>를 실행하거나 pycharm에서 수동으로 설치해준다.
필요한 module을 설치를 했는데도 import 오류가 난다면 library를 삭제 후 다시 설치한다.
버전이 올바른지 확인한다. 버전이 너무 높아도 오류가 날 수 있다.

- colaboratory 에서 구현

image transfer를 실행할 때, jupyter notebook에서 1000번을 돌리는데 너무 느리다면, colatboratory 에서 gpu를 사용하여 코드를 돌린다. 혹은 iteration을 줄인다.
image transfer가 잘 진행되는지 보고싶다면, 일정한 iteration마다(ex. 100 iteration) 이미지를 저장하도록

하여 경과를 관찰한다.

- 이미지 합성 결과에서 합성 결과가 이상할 때
image transfer의 변수설정에서 NOISE_RATIO, BETA, ALPHA의 변수의 값을 조정해준다.

6) 주석 포함 소스 코드

github 주소 : https://github.com/gksthd1992/gajah