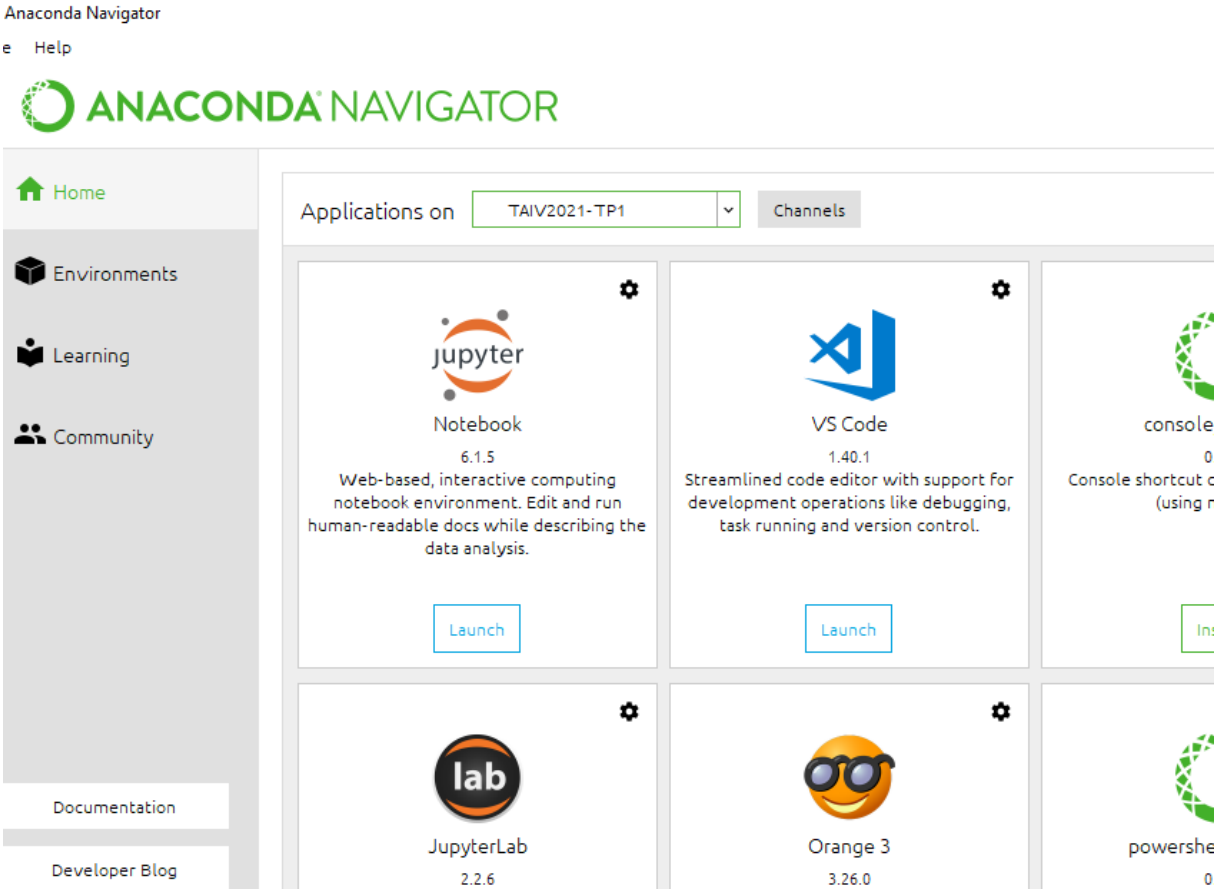
Master 2.Informatique Visuelle
Module Traitement et Analyse d'Images et de Vidéos
Année 2020/2021
-TP4-

Activer l'environnement du TP

```
(base) C:\Users\Utilisateur>conda activate TAIV2021-TP1
```

Sous Anaconda, il suffit de lancer jupyter à partir de l'interface après avoir choisi l'environnement du TP :



Partie1

```
importnumpy as np
import cv2 as cv
importmatplotlib.pyplot as plt
img = cv.imread('car.jpg',0)

plt.imshow(img)

ret,thresh1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
ret,thresh2 = cv.threshold(img,127,255,cv.THRESH_BINARY_INV)
ret,thresh3 = cv.threshold(img,127,255,cv.THRESH_TRUNC)
ret,thresh4 = cv.threshold(img,127,255,cv.THRESH_TOZERO)
ret,thresh5 = cv.threshold(img,127,255,cv.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
fori in range(6):
plt.subplot(2,3,i+1),plt.imshow(images[i],'gray',vmin=0,vmax=255)
plt.title(titles[i])
plt.xticks([]),plt.yticks([])
```

Partie2

```
import cv2 as cv
importnumpy as np
frommatplotlib import pyplot as plt
img = cv.imread('car.jpg',0)
# globalthresholding
ret1,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
# Otsu's thresholding
ret2,th2 = cv.threshold(img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
# Otsu's thresholding after Gaussian filtering
#blur = cv.GaussianBlur(img,(5,5),0)
ret3,th3 = cv.threshold(img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
# plot all the images and their histograms
images = [img, 0, th1,
img, 0, th2,
img, 0, th3]
titles = ['Original Noisy Image','Histogram','GlobalThresholding (v=127)',
      'Original Noisy Image','Histogram',"Otsu'sThresholding",
      'Gaussian filtered Image','Histogram',"Otsu'sThresholding"]
fori in range(3):
plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray')
plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'gray')
plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```

Partie 3

```
from __future__ import print_function
import cv2
importnumpy as np
importmatplotlib.pyplot as plt
import random as rng
%matplotlib

def display(pic,cmap='gray'):
```

```
fig = plt.figure(figsize=(12,10))
myplot = fig.add_subplot(111)
myplot.imshow(pic,cmap='gray')

img = cv2.imread('appleorange.jpg')
#img.shape

plt.imshow(img)

display(img)

threshold =100
canny_output = cv2.Canny(img, threshold, threshold * 2)
contours, hierarchy = cv2.findContours(canny_output, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

print(len(contours))

count=0
font = cv2.FONT_HERSHEY_SIMPLEX
for c in contours:
x,y,w,h = cv2.boundingRect(c)
if (w >20 and h >20):
count = count+1
      ROI = img[y+int(h/4):y+int(3*h/4), x+int(h/4):x+int(3*h/4)]
ROI_meancolor = cv2.mean(ROI)
print(count,ROI_meancolor)
if (ROI_meancolor[0] > 30 and ROI_meancolor[0] < 40 and ROI_meancolor[1] > 70 and ROI_meancolor[1] < 105
andROI_meancolor[2] > 150 and ROI_meancolor[2] < 200):
cv2.putText(img, 'orange', (x-2, y-2), font, 0.8, (255,255,255), 2, cv2.LINE_AA)
cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,255),3)
cv2.imshow('Contours', img)
else:
cv2.putText(img, 'apple', (x-2, y-2), font, 0.8, (0,0,255), 2, cv2.LINE_AA)
cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
cv2.imshow('Contours', img)

fromskimage.feature import hog
fromskimage import data, exposure
fruit, hog_image = hog(img, orientations=8, pixels_per_cell=(16, 16),
cells_per_block=(1, 1), visualize=True, multichannel=True)
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
cv2.imshow('HOG_image', hog_image_rescaled)

cv2.imshow('HOG_image', hog_image_rescaled)
```

Partie4

```
import cv2
importnumpy as np
importmatplotlib.pyplot as plt
%matplotlib inline

cv2.__version__

def display(pic,cmap='gray'):
fig = plt.figure(figsize=(12,10))
myplot = fig.add_subplot(111)
myplot.imshow(pic,cmap='gray')

tile = cv2.imread('tile.jpg',0)
```

```
display(tile)

bathroom = cv2.imread('bathroom_image.jpg',0)

display(bathroom)

orb = cv2.ORB_create()
kp1,des1 = orb.detectAndCompute(tile,None)
kp2,des2 = orb.detectAndCompute(bathroom,None)

bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

matches = bf.match(des1,des2)

matches = sorted(matches,key=lambda x:x.distance)

floor_matches = cv2.drawMatches(tile,kp1,bathroom,kp2,matches[:50],None,flags=2)

display(floor_matches)

sift = cv2.xfeatures2d.SIFT_create()

kp1,des1 = sift.detectAndCompute(tile,None)
kp2,des2 = sift.detectAndCompute(bathroom,None)

bf = cv2.BFMatcher()

matches = bf.knnMatch(des1,des2,k=2)

good = []
for match1,match2 in matches:
if match1.distance < 1.2*match2.distance:
good.append([match1])

draw_params = dict(matchColor=(0,0,255),singlePointColor=(0,0,255))
sift_matches = cv2.drawMatchesKnn(tile,kp1,bathroom,kp2,good,None,**draw_params)

display(sift_matches)

sift = cv2.xfeatures2d.SIFT_create()

kp1,des1 = sift.detectAndCompute(tile,None)
kp2,des2 = sift.detectAndCompute(bathroom,None)

flann_index_kdtree=0
index_params = dict(algorithm = flann_index_kdtree,trees=7)
search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params,search_params)

matches = flann.knnMatch(des1,des2,k=2)

matchesMask = [[0,0] for i in range(len(matches))]


fori,(match1,match2) in enumerate(matches):
if match1.distance <1.2*match2.distance:
matchesMask[i]=[1,0]

draw_params = dict(matchColor=(255,0,0),singlePointColor=(0,0,255),matchesMask=matchesMask,flags=2)

flann_matches = cv2.drawMatchesKnn(tile,kp1,bathroom,kp2,matches,None,**draw_params)
```

```
display(flann_matches)
```

Travail à faire :

Comprendre le code,