Lab Instructions - session 3

Working with videos, histograms

Part 1: Reading and displaying videos

You can see a short video clip called “kntu-computer.avi”. The following code opens this files and displays it. You can also find the source file in your instructions folder, named “**read_video.py**”.

```
import numpy as np
import cv2

# create a VideoCapture object
cap = cv2.VideoCapture('kntu-computer.avi')

# sometimes this is needed:
#if not cap.isOpened():
#    cap.open();

while True:
    # Capture frame-by-frame
    ret, I = cap.read()

    if ret == False: # end of video (perhaps)
        break

    cv2.imshow('win1',I) # Display I

    key = cv2.waitKey(33) # ~ 30 frames per second

    if key == ord('q'): # exit when “q” is pressed
        break

    # replace the above with "if key 0xFF == ord('q')"
    # if it fails

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

- What happens by pressing "q" before the video finishes? (replace "`key == ord('q')`" by "`key && 0xFF == ord('q')`" if the above fails.
- `key = cv2.waitKey(33)` creates a delay of 33 milliseconds. What happens if you increase or decrease this value? Change it to 3 or 300 and see what happens.
- replace `cv2.waitKey(33)` by `cv2.waitKey(0)` or `cv2.waitKey()` and see what happens.

Writing a video on the disk

Open the file “lab3_task1.py”. It reads a video file named **“eggs.avi”** and saves the frames into another video file named **“eggs-reverse.avi”**. Run the file.

```
import numpy as np
import cv2

# create a VideoCapture object
cap = cv2.VideoCapture('eggs.avi')

# get the dimensions of the frame
# you can also read the first frame to get these
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  # width of the frame
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) # height of the frame

fourcc = cv2.VideoWriter_fourcc(*'XVID') # choose codec

# opencv 2.x:
#w = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_WIDTH))
#h = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT))
#fourcc = cv2.cv.CV_FOURCC(*'XVID')

# create VideoWriter object w by h, 30 frames per second
out = cv2.VideoWriter('eggs-reverse.avi',fourcc, 30.0, (w,h))

while True:
    ret, I = cap.read()

    if ret == False: # end of video (or error)
        break

    # write the current frame I
    out.write(I)

cap.release()
out.release()
```

Task 1: Change the above file so that the video frames are saved in reverse order. Therefore, at the end, the file `'eggs-reverse.avi'` should be a backward playback of ‘eggs.avi’. You can use python lists for buffering the frames if you need to:

```
buffer = []
while True:
    ...
    buffer.append(I) # add frame I at the end of the buffer
```

Part 2: Histograms

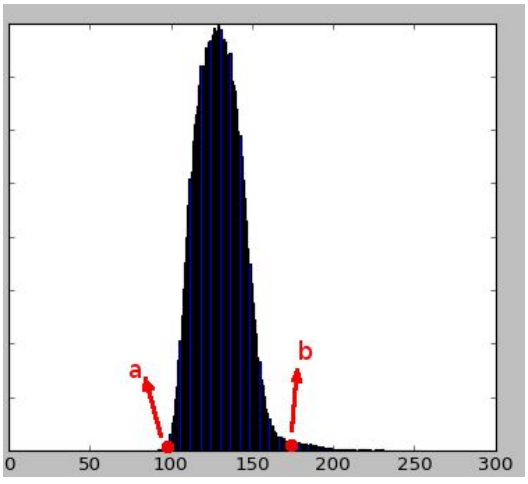
Here, we use matplotlib (not OpenCV) to plot histograms. Open the file **lab3_task2.py**.

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

fname = 'crayfish.jpg'
#fname = 'office.jpg'

I = cv2.imread(fname, cv2.IMREAD_GRAYSCALE)

f, axes = plt.subplots(2, 3)

axes[0,0].imshow(I, 'gray', vmin=0, vmax=255)
axes[0,0].axis('off')

axes[1,0].hist(I.ravel(),256,[0,256]);

plt.show()
```

`plt.subplots(2,3)` creates a 2 by 3 array of subplots (2 rows, 3 columns). By running the above code, you can see that only the first column of the subplots are used (axes[0,0] and axes[1,0]). The image is plotted in axes[0,0] and its histogram in axes[1,0].

- What does **I.ravel()** do in the above? Why has it been used?

Task 2: (a) We want to linearly expand the histogram to get a better contrast. Determine points **a** and **b** for linear histogram expansion according to the image below:



Now, create an image **J** in which the histogram has been expanded. You may use the following piece of code.

```
J = (I-a) * 255.0 / (b-a)
J[J < 0] = 0
J[J > 255] = 255
J = J.astype(np.uint8)
```

plot the image **J** and its histogram in the second column of the subplots (axes[0,1] and axes[1,1]).

- What does the above piece of code do?

Task 2: (b) You can perform histogram equalization in OpenCV using the following function.

```
K = cv2.equalizeHist(I)
```

plot the image **K** and its histogram in the third column of the subplots (axes[0,2] and axes[1,2]).

- Compare the linearly histogram expanded image with the histogram equalization.
- Do this for a bunch of other images in your folder (crayfish.jpg, map.jpg, train.jpg, branches.jpg, terrain.jpg). Note that you need to change **a** and **b** for each image.

Extra score (1 point)

- Can you think of a way of automatically obtaining **a** and **b** in Task2(a) for arbitrary images? Implement it.