Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

Problem description

This assignment presented basic or primary computer vision challenges when it comes to image/video processing. The challenges consisted of taking a video and interact with the frames to produce an output. Such challenges consisted of:

1. Transforming or changing the image to grayscale
2. Producing an inversion of the grayscale image, or a negative grayscale image
3. Manipulating the image to extract a mirror and upside-down effect
4. Mapping high and low-density colors to modify the gamma of the image
5. Using user input to record specific frames given the input
6. Elaborate a method to detect motion

Algorithms implemented

In this section the problems above will be decomposed into their core functionality or implementation. All implementations were built by researching the documentation of OpenCV. Refer to the section Discussion of results to read the explanation and logic.

Problem 1 – Grayscale

Algorithms: `cvtColor(imageToChange, cv2.COLOR_BGR2GRAY)`

Problem 2 - Inverse Grayscale

Algorithms:

```
cv2.imshow("frame", 1 - gray) #current range is gray, thus [1 - gray]
```

Problem 3 – Mirror Image

Algorithms:

```
mirror = cv2.flip(frame, 1) # 1 flips the image on the Y - axis
```

Problem 4 – Upside-down image

Algorithms:

```
upside_down = cv2.flip(frame, 0) # passing argument [0] flips the image
#given the X - axis
```

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

Problem 5 – User-defined variable for frame capture

Algorithm:

```
n = int(input("Please type a number from 1 to 10\n"))

out = cv2.VideoWriter('outpy.mp4',cv2.VideoWriter_fourcc('M','J','P','G'),
24, (640,480))

#inside the loop

if int(time.time() - start) % n == 0:

        out.write(frame)

#outside the loop

out = release()
```

Problem 6 - Illumination Correction

****NOT IMPLEMENTED****

Problem 7 – Motion Tracker

Algorithms:

```
motion_detector = cv2.createBackgroundSubtractorMOG2(120, 100.0,False)

while True:

    ret,frame=cap.read()

    #applied a grayscale to reduce noise

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    detector = motion_detector.apply(gray)

    cv2.imshow('MotionDetector',detector)
```

…

Experimental results

Problem 1

By using cv2.imshow(“frame”, frame[:,:,1]) , the image shown is transformed into a grayscale. I am not really sure how this works, however, I believe it is only using one dimension of colors.

Motion Tracker

Changing the image to grayscale appears to reduce some noise, also the shadows may cause some noise.

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

Discussion of results

Problem 1

OpenCV incorporates methods that can read an image and modify the values of such images. One of the most commonly used functions is to transform a certain input (image) from color to grayscale. The first parameter takes an image as the input and the second parameter is the transformation that the image will go through. This function returns the image after the transformation.

Problem 2

From problem 1, an image was transformed into a grayscale image. Taking that into account for this problem, the next step is to produce and inversion of the grayscale and output such image with the inversion applied. My solution was to subtract the grayscale range from a positive integer.

The reasoning behind this comes from set notation. If there exists a set in which all positive colors are held, then there is an inverse of such set that the negative colors are held. The inverse can be obtain using the following formula: 1 – [range].

Problem 3

For this problem I researched in depth about linear transformations and different operations given a matrix. My understanding of computer vision is that an image is represented as a matrix of pixels, each pixel contains an RGB value. With this information, I planned to produce an axis flip on the image to reproduce a "mirror" effect. Using OpenCV documentation, I found a function that flips the input graphic and returns it as output. The function is called **flip(image, flipParameter)**. The flip parameter is an integer from -1 to 1 and each integer in that range represents an axis flip. -1 flips the image in both axes, 0 flips the image in the X – axis, and 1 flips the image in the Y – axis.

Problem 4

Using the same function from problem 3, I changed the second parameter of the function **flip(image, flipParameter).**

Problem 5

In this problem I went to the OpenCV documentation and figured out that I needed the following requirements:

```
VideoWriter, FourCC code, FPS, Frame Size
```

The VideoWriter handles the output file that is going to be produced, it takes in a FourCC value, a frame rate for the output video, and a size for the window or video. The next lines show how I managed to solve this problem.

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes
To take the user input I used the input() function and casted it to int.

```
n = int(input("Please type a number from 1 to 10\n"))
```

Next, I created a variable name “out” that will hold the output video

```
out =
cv2.VideoWriter('outpy.mp4',cv2.VideoWriter_fourcc('M','J','P','G'), 24,
(640,480))
```

To take a frame each n seconds provided by the user I used the formula that was originally in the code for counting time. Then, I took the time formula and get the modulus result from the n variable that the user supplied.

Let’s say the n variable was 2, this means that every 2 seconds the program will capture a frame and write it to a file. So, for every second that passes the algorithm I used would take the modulus of the time and the n variable, like this:

```
if int(time.time() - start) % n == 0:
        out.write(frame)
```

If the condition is true, it means that 2 seconds had passed, then just write the frame to the output file. After exiting the loop, it is important to close the output file. This is done using:

```
out = release()
```

Problem 6

More research on this topic

Related topics: Gamma, Hue, HSV values, Brightness, Contrast, Morphological transformation.

Problem 7

In this problem I used a background subtraction algorithm from OpenCV. It is called the Gaussian Mixture-based Background/Foreground Segmentation Algorithm. The way this algorithm works is by using K Gaussian distributions. It basically compares pixels that are static with those that are not static. The not static pixels will be shown as with pixels, whereas the static pixels are shown as black pixels.

The comparison uses multiple images layered to form a mask and compares pixels. With this algorithm, it can be seen which pixels are static and non-static.

This function contains optional parameters. History, n gaussian mixtures, a threshold, and a shadow Boolean. Refer to the Appendix to see the code.

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

Conclusions

Computer Vision can work with minimal errors if the algorithm used is efficient and fast. The integration of loops slows down the process of image processing. OpenCV is a great tool that contains many functions that perform different operations in real-time analysis on images. Although, I encountered some challenges to complete some problems, I feel I can research more about alternative solutions, rather than using functions. In other words, I would like to know more, so that I can make up my own approximations to the real solutions.

Appendix

```
import numpy as np

import cv2

import time


#input from user to get the amount of time before each frame is written to another file

#no excpetion handling implemented

#n = int(input("Please type a number from 1 to 10\n"))


cap = cv2.VideoCapture(0)

start = time.time()

count=0


#For part 5 of Lab1

#out = cv2.VideoWriter('outpy.mp4',cv2.VideoWriter_fourcc('M','J','P','G'), 24, (640,480))

while(True):

    # Capture frame-by-frame

    ret, frame = cap.read()


    count+=1


    cv2.imshow('frame0',frame)
```

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

```
    #----------1. Display the gray level version of the image--------------#

    #this modifies the frame capture to grayscale

    #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #cv2.imshow("frame", gray)




    #--------2. Display the negative gray level version of the image--------#

    #color range in frames is [255], inverse is [1 – range]

    #From statistics, set A contains all the colors, then set A' contains the
not-colors or negative of colors

    #cv2.imshow("frame", 1 - gray) #current range is gray, thus [1 - gray]




    #-------3. Display the mirrored version of the original color image-----#

    #mirror = cv2.flip(frame, 1) # passing argument [1] flips the image given
#Y - axis

    #cv2.imshow("frame", mirror)


    #------4. Display the original color image upside down-------------------
#

    #upside_down = cv2.flip(frame, 0) # passing argument [0] flips the image
#given the X - axis

    #cv2.imshow("frame", upside_down)


    #---5. Write to a file one frame every n second, where n is a user-
supplied parameter.

    #open VideoWriter

    #out = cv2.VideoWriter('myVideo.avi',
#cv2.VideoWriter_fourcc('M','J','P','G'),24, Size(frame_width, frame_height))

   #IMPORTANT: typecast time to int to make the condition true given time % n

   # if int(time.time() - start) % n == 0:

   #     out.write(frame)


    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

    if count%30==0:
```

```
        print(np.max(frame),np.min(frame))


elapsed_time = time.time()-start

print('Capture speed: {0:.2f} frames per second'.format(count/elapsed_time))


# When everything done, release the capture

cap.release()

#to release VideoWriter

#out = release()

cv2.destroyAllWindows()
```

MOTION TRACKER

```
import cv2

#OpenCV Documentation
https://docs.opencv.org/3.3.1/de/de1/group__video__motion.html

cap = cv2.VideoCapture(0)

motion_detector = cv2.createBackgroundSubtractorMOG2(120, 100.0,False)


while True:

    ret,frame=cap.read()

    #applied a grayscale to reduce noise

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    detector = motion_detector.apply(gray)


    cv2.imshow('MotionDetector',detector)


    if cv2.waitKey(1) & 0xFF == ord('q'):

        break


cap.release()

cv2.destroyAllWindows()
```

Sebastian Gonzalez
Lab 1
Computer Vision - Dr. Olac Fuentes

References

Grayscale function

Getting Started with Images. Retrieved September 9, 2018 from https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html

VideoCapture method

Getting Started with Videos. Retrieved September 9, 2018 from https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html

Flip method

Operations on Arrays. Retrieved September 9, 2018 from https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#flip

Background Subtraction Section 1.6.3 in PDF

Alexander Mordvintsev. 2017. OpenCV-Python Tutorials Documentation. (November 2017). Retrieved September 9, 2018 from https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf

Motion Detector

OpenCV: Motion Analysis. Retrieved September 9, 2018 from https://docs.opencv.org/3.3.1/de/de1/group__video__motion.html

Thresholds

Thresholding OpenCV Python Tutorial. Retrieved September 9, 2018 from https://pythonprogramming.net/thresholding-image-analysis-python-opencv-tutorial/

https://pythonprogramming.net/thresholding-image-analysis-python-opencv-tutorial/