**CSC 745 Advanced Multimedia Programming**

**Exercise: Image processing with OpenCV**

**Goals**

Learn the basics of OpenCV, an open-source computer vision library with Python bindings.

**Background**

OpenCV was initially developed by Intel, beginning in 1999. The primary goals of the project were to advance both computer vision research and vision-based commercial applications. Written in C++ OpenCV has interfaces for C++, Python, Java, and Matlab and runs on Windows, Linux, Android, and MacOS. Unlike most image systems, OpenCV stores images in BGR format, rather than RGB.

**Procedure**

1. Download the exercise folder from Blackboard and unzip it.
2. Open an Anaconda prompt, create a virtual environment, and install the latest version of OpenCV:

conda create -n cv Python=3.11

conda activate cv

pip install opencv-python #I had no luck with conda install opencv

pip install opencv-contrib-python

1. Test your installation (be sure your cv environment is still active):

#Open a Python shell – for example, open a shell in idle

import cv2 #The OpenCV module is imported as cv2 so programs are backward compatible

print(cv2.\_\_version\_\_)

1. Load and display an image, and return it:

img = cv2.imread(fname)

print(type(img)) #Confirm the type of the image

print(img.shape) #img is an np.ndarray; print the rows, columns, and number of color channels

cv2.imshow(“Loaded Image”, img) #Title of window and image object

cv2.waitKey(0) #Wait for a keypress

cv2.destroyAllWindows() #Close the display window

return img

1. Convert the image to various color spaces. Grayscale conversion is shown; also try HSV (BGR2HSV), YCrCb (BGR2YCrCb), CIE LAB (BGR2LAB), and CIE LUV (BGR2LUV). Most of these images will look strange when you display them because they will be displayed as though they are BGR) – but the three channels no longer correspond to those colors.

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2.imshow(“Grayscale Image”, gray\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

1. Blur the image.:

blurred\_img = cv2.GaussianBlur(img, (21, 21), 0)

cv2.imshow(“Blurred Image”, blurred\_img)

cv2.waitKey(0)

cv2.destroyAllWindows

1. Create an edge image. There are many algorithms for extracting edges. Use the Canny edge detector. Hopefully we’ll have a chance to talk about these algorithms in class at some point.

edge\_img = cv2.Canny(img, 100, 200) #Args are lower and upper thresholds for hysteresis

#Display the edge image as usual

1. Modify a pixel. Because the image is an np array, the indices are [row, column]. Modify a copy so the original is not changed. There are drawing methods that can be used to draw a line – but the goal here is to learn to modify individual pixels.

copy = img.copy()

copy[95:105, 45:55] = [255, 0, 0] #Set a small square of pixels to blue

cv2.imshow(“Modified Image”, copy)

extraction = copy[75:125, 25:75] #Extract a small area around the modified pixels

cv2.imshow(“Extracted Image”, extraction) #Display the extracted region

cv2.waitKey(0)

height, width, channels = copy.shape #Get the image dimensions

mid\_col = width//2 #Find the middle column

copy[:, mid\_col] = [0, 0, 255] #Draw a vertical red line down the center

#Or, to draw a thicker line: copy[:, mid\_col-5:mid\_col+5] = [0, 0, 255]

#Add a green horizontal line across the middle of the image

cv2.imshow(“Modified Image”, copy)

cv2.destroyAllWindows()

1. Open and play a video. This will only play the image frames – not the audio. Playing the audio will require using another package to read the audio frames in synchronization with OpenCV’s reading the images frames.

cap = cv2.VideoCapture(fname) #Create a capture object

if cap.isOpened():

while True:

ret, frame = cap.read() #Read the return value and the frame

if not ret:

break #Break the loop when the video ends

cv2.imshow("Video Frame", frame)

if cv2.waitKey(25) & 0xFF == ord('q'):

break #Quit the video on 'q' key

else:

print("Error: Unable to open the video file")

cap.release()

cv2.destroyAllWindows()

1. Video compression makes use of frame to frame redundancy. That raises the question – how similar are successive frames in a video? Here we will illustrate the frame to frame similarity by displaying the difference between video frames. We’ll use the ‘d’ key to toggle between displaying the frames and the frame differences.

show\_diff = False #Determines whether to display the frame or the diff

cap = cv2.VideoCapture(fname)

if cap.isOpened:

ret, prev\_frame = cap.read() #Read first frame

while True:

ret, frame = cap.read()

if not ret:

break #Break loop when video ends

if show\_diff:

frame\_diff = cv2.absdiff(prev\_frame, frame)

cv2.imshow("Video Frame", frame\_diff)

else:

cv2.imshow("Video Frame", frame)

prev\_frame = frame

key = cv2.waitKey(25) #Wait 25 ms for a key press

if key & 0xFF == ord('d'):

if show\_diff:

show\_diff = False

else:

show\_diff = True

elif key & 0xFF == ord('q'):

break #Quit the video on 'q' key

else:

print("Error: Unable to open the video file")

cap.release()

cv2.destroyAllWindows()

There are many OpenCV tutorials and courses on the Internet. And, of course, you can read the documentation to find more information about OpenCV’s capabilities.

**Deliverables**

Submit your .py file on Blackboard before the due date.