EDGE FINDING AND CORNER DETECTION:import cv2

a.**Edge detection OpenCV python**

```
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path) #reading the image
edges = cv2.Canny(img,100,200) #canney edhe detecton
cv2.imshow('Edges in the image', edges) #displaying the image
cv2.waitKey(0)
```

b.Canny Edge Detection in OpenCV

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv.imread(path)
edges = cv.Canny(img,100,200)
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
```

Sobel Edge Detection and Canny Edge Detection

```
import cv2
# Read the original image
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path)
# Display original image
cv2.imshow('Original', img)
cv2.waitKey(0)
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel
Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel
Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) #
Combined X and Y Sobel Edge Detection
# Display Sobel Edge Detection Images
cv2.imshow('Sobel X', sobelx)
cv2.waitKey(0)
cv2.imshow('Sobel Y', sobely)
cv2.waitKey(0)
cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
cv2.waitKey(0)
# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge
Detection
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

How to use K-means clustering for image segmentation using OpenCV in python

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# read the image
```

```
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
image = cv2.imread(path)
# convert to RGB
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
# reshape the image to a 2D array of pixels and 3 color values (RGB)
pixel_values = image.reshape((-1, 3))
# convert to float
pixel_values = np.float32(pixel_values)
print(pixel_values.shape)
# define stopping criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
# number of clusters (K)
k = 3
_, labels, (centers) = cv2.kmeans(pixel_values, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)
# convert back to 8 bit values
centers = np.uint8(centers)
# flatten the labels array
labels = labels.flatten()
# convert all pixels to the color of the centroids
segmented_image = centers[labels.flatten()]
# reshape back to the original image dimension
segmented_image = segmented_image.reshape(image.shape)
# show the image
plt.imshow(segmented_image)
plt.show()
# disable only the cluster number 2 (turn the pixel into black)
masked_image = np.copy(image)
# convert to the shape of a vector of pixel values
```

```
masked_image = masked_image.reshape((-1, 3))
# color (i.e cluster) to disable
cluster = 2
masked_image[labels == cluster] = [0, 0, 0]
# convert back to original shape
masked_image = masked_image.reshape(image.shape)
# show the image
plt.imshow(masked_image)
plt.show()
```

Black and white image colorization with OpenCV and Deep Learning Source Code:

```
import numpy as np
import cv2
from cv2 import dnn
#--------Model file paths--------#
proto_file =
r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\colorization_deploy_v2.prototxt'
model_file =
r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\colorization_release_v2.caffemod
el'
hull_pts = r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\pts_in_hull.npy'
img_path = r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\GG.jpg'
#--------------#--------------#
#--------Reading the model params--------#
net = dnn.readNetFromCaffe(proto_file,model_file)
kernel = np.load(hull_pts)
#-----------------------------------#--------------------#
#-----Reading and preprocessing image--------#
img = cv2.imread(img_path)
```

```
scaled = img.astype("float32") / 255.0
lab_img = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)
#-----------------------------------#--------------------#
# add the cluster centers as 1x1 convolutions to the model
class8 = net.getLayerId("class8_ab")
conv8 = net.getLayerId("conv8_313_rh")
pts = kernel.transpose().reshape(2, 313, 1, 1)
net.getLayer(class8).blobs = [pts.astype("float32")]
net.getLayer(conv8).blobs = [np.full([1, 313], 2.606, dtype="float32")]
#-----------------------------------#--------------------#
# we'll resize the image for the network
resized = cv2.resize(lab_img, (224, 224))
# split the L channel
L = cv2.split(resized)[0]
# mean subtraction
L -= 50
#-----------------------------------#--------------------#
# predicting the ab channels from the input L channel
net.setInput(cv2.dnn.blobFromImage(L))
ab_channel = net.forward()[0, :, :, :].transpose((1, 2, 0))
# resize the predicted 'ab' volume to the same dimensions as our
# input image
ab_channel = cv2.resize(ab_channel, (img.shape[1], img.shape[0]))
# Take the L channel from the image
L = cv2.split(lab_img)[0]
# Join the L channel with predicted ab channel
colorized = np.concatenate((L[:, :, np.newaxis], ab_channel), axis=2)
# Then convert the image from Lab to BGR
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
```

```
colorized = np.clip(colorized, 0, 1)
# change the image to 0-255 range and convert it from float32 to int
colorized = (255 * colorized).astype("uint8")
# Let's resize the images and show them together
img = cv2.resize(img,(640,640))
colorized = cv2.resize(colorized,(640,640))
result = cv2.hconcat([img,colorized])
cv2.imshow("Grayscale -> Colour", result)
cv2.waitKey(0)
```

Black and white video colorization with OpenCV and Deep Learning Source Code:

```
import numpy as np
import cv2
from cv2 import dnn
import imutils
import os
from os import listdir
from os.path import join
#--------Model file paths--------#
prototxt =
r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\colorization_deploy_v2.prototxt'
model =
r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\colorization_release_v2.caffemod
el'
points = r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\pts_in_hull.npy'
video = r'C:\Users\WAINAINA\Desktop\VISION\CODES\LESSON5\GEORGE.mp4'
width = 500
vs = cv2.VideoCapture(video)
#--------------#--------------#
```

```
#--------Reading the model params--------#
nnet = cv2.dnn.readNetFromCaffe(prototxt,model)
pts = np.load(points)
class8 = nnet.getLayerId("class8_ab")
conv8 = nnet.getLayerId("conv8_313_rh")
ts = pts.transpose().reshape(2, 313, 1, 1)
nnet.getLayer(class8).blobs = [pts.astype("float32")]
nnet.getLayer(conv8).blobs = [np.full([1, 313], 2.606, dtype="float32")]
#----------------------------------#--------------------#
count = 0
success = True
while success:
success, frame = vs.read()
if frame is None:
break
frame = imutils.resize(frame, 500)
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
frame = cv2.cvtColor(frame, cv2.COLOR_GRAY2RGB)
scaled = frame.astype("float32") / 255.0
lab = cv2.cvtColor(scaled, cv2.COLOR_RGB2LAB)
resized = cv2.resize(lab, (224, 224))
L = cv2.split(resized)[0]
L -= 50
nnet.setInput(cv2.dnn.blobFromImage(L))
ab = nnet.forward()[0, :, :, :].transpose((1, 2, 0))
ab = cv2.resize(ab, (frame.shape[1], frame.shape[0]))
L = cv2.split(lab)[0]
colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
```

```
colorized = np.clip(colorized, 0, 1)
colorized = (255 * colorized).astype("uint8")
cv2.imshow("Original", frame)
cv2.imshow("Colorized", colorized)
cv2.imwrite("./colorized_video_frames/frame%d.jpg" % count, colorized)
count += 1
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
break
vs.release()
cv2.destroyAllWindows()
def convert_frames_to_video(pathIn, pathOut, fps):
frame_array = []
files = [f for f in os.listdir(pathIn) if os.isfile(join(pathIn, f))]
#for sorting the file names properly
files.sort(key = lambda x: int(x[5:-4]))
for i in range(len(files)):
filename=pathIn + files[i]
#reading each files
img = cv2.imread(filename)
height, width, layers = img.shape
size = (width,height)
print(filename)
#inserting the frames into an image array
frame_array.append(img)
out = cv2.VideoWriter(pathOut,cv2.VideoWriter_fourcc(*'MJPG'), fps, size)
for i in range(len(frame_array)):
# writing to a image array
out.write(frame_array[i])
```

```
out.release()
pathIn= './colorized_video_frames/'
pathOut = './colorized_videos/video.avi'
fps = 30.0
convert_frames_to_video(pathIn, pathOut, fps)
```

Crop Images

We can also crop subimages with the slicing function. We crop the image from (90, 50), i.e. row 90 and column 50, to (50, 120) in the following example:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import numpy as np
def imag_tile(img, n, m=1):
"""
The image "img" will be repeated n times in
vertical and m times in horizontal direction.
"""
if n == 1:
tiled_img = img
else:
lst_imgs = []
for i in range(n):
lst_imgs.append(img)
tiled_img = np.concatenate(lst_imgs, axis=1 )
if m > 1:
lst_imgs = []
for i in range(m):
lst_imgs.append(tiled_img)
```

```
tiled_img = np.concatenate(lst_imgs, axis=0 )
return tiled_img
basic_pattern = cv2.imread (r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg')
decorators_img = imag_tile(basic_pattern, 2, 2)
cropped = basic_pattern[90:150, 50:120]
plt.axis("off")
plt.imshow(cropped)
cv2.imshow('MY OUTPUT',cropped)
cv2.waitKey(0)
```

Image Augmentation

a.To Flip an Image Horizontally

To flip an image horizontally (along the image's vertical axis), we use the following code shown below.

```
import cv2
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
# Using cv2.imread() method
image = cv2.imread(path)
flippedimage= cv2.flip(image, 1)
cv2.imshow('Horizontally Flipped Image', flippedimage)
cv2.waitKey(0)
```

b.To Flip an Image Vertically

To flip an image vertically (along the image's horizontal axis), we use the following code shown below.

```
import cv2
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
# Using cv2.imread() method
image = cv2.imread(path)
flippedimage= cv2.flip(image, 0)
cv2.imshow('Vertically Flipped Image', flippedimage)
```

```
cv2.waitKey(0)
```

c.To Flip an Image Horizontally and Vertically

To flip an image horizontally and vertically, we use the following code shown below.

```
import cv2
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
# Using cv2.imread() method
image = cv2.imread(path)
flippedimage= cv2.flip(image, -1)
cv2.imshow('Vertically Flipped Image', flippedimage)
cv2.waitKey(0)
```

d.Rotating

The rotate() method of Python Image Processing Library Pillow Takes the number of degrees as a parameter and rotates the image in Counter Clockwise Direction to the number of degrees specified.

```
# import the Python Image processing Library
import imutils
import cv2
# Create an Image object from an Image
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
image = cv2.imread(path)
Rotated_image = imutils.rotate(image, angle=45)
Rotated1_image = imutils.rotate(image, angle=90)
# display the image using OpenCV of
# angle 45
cv2.imshow("Rotated 45 Degrees", Rotated_image)
# display the image using OpenCV of
# angle 90
```

```
cv2.imshow("Rotated 90 Degrees", Rotated1_image)
# This is used for To Keep On Displaying
# The Image Until Any Key is Pressed
cv2.waitKey(0)
cv2.waitKey(0)
```

Fourier Transform

is like blurring BT more advanced .

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path,0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
import numpy as np
import cv2
from matplotlib import pyplot as plt
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path,0)
dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))
```

```
rows, cols = img.shape
crow,ccol = rows/2 , cols/2
# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[int(crow-30):int(crow+30), int(ccol-30):int(ccol+30)] = 1
# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])
plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

Convolutional neural networks.

Introduction

In computer vision, we have a convolutional neural network that is very popular for computer vision tasks like image classification, object detection, image segmentation and a lot more.

A python Program to return the height, weight and RGB of the image and rotate it:

```
# Importing the OpenCV library
import cv2
# Reading the image using imread() function
path = r'C:\Users\WAINAINA\Desktop\VISION\WAINAINA.jpg'
img = cv2.imread(path)
# Extracting the height and width of an image
h, w = img.shape[:2]
# Displaying the height and width
```

```
print("Height = {}, Width = {}".format(h, w))
# Extracting RGB values.
# Here we have randomly chosen a pixel by passing in 100, 100 for height and
width.
(B, G, R) = img[100, 100]
# Displaying the pixel values
print("R = {}, G = {}, B = {}".format(R, G, B))
B = img[100, 100, 0]
print("B = {}".format(B))
# resize() function takes 2 parameters, the image and the dimensions
resize = cv2.resize(img, (800, 800))
# Calculating the ratio
ratio = 800 / w
# Creating a tuple containing width and height
dim = (800, int(h * ratio))
# Resizing the image
resize_aspect = cv2.resize(img, dim)
# Calculating the center of the image
center = (w // 2, h // 2)
# Generating a rotation matrix
matrix = cv2.getRotationMatrix2D(center, -45, 1.0)
# Performing the affine transformation
rotated = cv2.warpAffine(img, matrix, (w, h))
cv2.imshow('resize image',rotated)
# We are copying the original image, as it is an in-place operation.
output = img.copy()
cv2.waitKey(0)
```

People detection

This method is trained to detect pedestrians, which are human mostly standing up, and fully visible.

pip install imutils - A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

A) **Detect from webcam**:

```
# import the necessary packages
import numpy as np
import cv2
# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
cv2.startWindowThread()
# open webcam video stream
cap = cv2.VideoCapture(0)
# the output will be written to output.avi
out = cv2.VideoWriter(
'output.avi',
cv2.VideoWriter_fourcc(*'MJPG'),
15.,
(640,480))
while(True):
# Capture frame-by-frame
ret, frame = cap.read()
# resizing for faster detection
frame = cv2.resize(frame, (640, 480))
# using a greyscale picture, also for faster detection
gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
# detect people in the image
# returns the bounding boxes for the detected objects
```

```
boxes, weights = hog.detectMultiScale(frame, winStride=(8,8) )
boxes = np.array([[x, y, x + w, y + h] for (x, y, w, h) in boxes])
for (xA, yA, xB, yB) in boxes:
# display the detected boxes in the colour picture
cv2.rectangle(frame, (xA, yA), (xB, yB),
(0, 255, 0), 2)
# Write the output video
out.write(frame.astype('uint8'))
# Display the resulting frame
cv2.imshow('frame',frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
# When everything done, release the capture
cap.release()
# and release the output
out.release()
# finally, close the window
cv2.destroyAllWindows()
cv2.waitKey(1)
```

B) Detect from Image:

```
import cv2
import imutils
# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
# Reading the Image
path = r'C:\Users\WAINAINA\Desktop\VISION\H3.jpg'
```

```
image = cv2.imread(path)
# Resizing the Image
image = imutils.resize(image,
width=min(400, image.shape[1]))
# Detecting all the regions in the
# Image that has a pedestrians inside it
(regions, _) = hog.detectMultiScale(image,
winStride=(4, 4),
padding=(4, 4),
scale=1.05)
# Drawing the regions in the Image
for (x, y, w, h) in regions:
cv2.rectangle(image, (x, y),
(x + w, y + h),
(0, 0, 255), 2)
# Showing the output Image
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

c) Detect from Video:

```
import cv2
import imutils
# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
path = r'C:\Users\WAINAINA\Desktop\VISION\GEORGE.mp4'
cap = cv2.VideoCapture(path)
```

```
while cap.isOpened():
# Reading the video stream
ret, image = cap.read()
if ret:
image = imutils.resize(image,
width=min(400, image.shape[1]))
# Detecting all the regions
# in the Image that has a
# pedestrians inside it
(regions, _) = hog.detectMultiScale(image,
winStride=(4, 4),
padding=(4, 4),
scale=1.05)
# Drawing the regions in the
# Image
for (x, y, w, h) in regions:
cv2.rectangle(image, (x, y),
(x + w, y + h),
(0, 0, 255), 2)
# Showing the output Image
cv2.imshow("Image", image)
if cv2.waitKey(25) & 0xFF == ord('q'):
break
else:
break
cap.release()
cv2.destroyAllWindows()
```