**Task 2**: Cropping, flipping and dataset of live video feed

**Code**:

1. **Cropping**-

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

count = 0

while True:

ret, frame = cap.read()

rows, cols, ch = frame.shape

roi = frame[rows//2-50:rows//2+50, cols//2-50:cols//2+50]

if count%2==0:

cv2.imshow('frame', frame)

count += 1

else:

cv2.imshow('frame', roi)

count += 1

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

cap.release()

cv2.destroyAllWindows()

2. **Dataset**-

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

count = 1

while True:

ret, frame = cap.read()

cv2.imshow('frame', frame)

cv2.imwrite("Img%d.jpg" %count, frame)

cv2.waitKey(1)

count += 1

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

cap.release()

cv2.destroyAllWindows()

3. **Flipping every 4 seconds**-

import cv2

import numpy as np

import time

cap = cv2.VideoCapture(0)

start = time.time()

while True:

ret, frame = cap.read()

cv2.imshow('frame', frame)

if int(time.time()-start)%4==0:

vnflip = cv2.flip(frame, 0)

cv2.imshow('frame', vnflip)

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

cap.release()

cv2.destroyAllWindows()

**Meaning**:

1) The live video feed is cropped using the “region of image/interest” concept and then shown alternately.

2) A dataset consists of every frame captured individually. Hence, each frame is showed and then subsequently saved using the cv2.imwrite() function.

3) The time.time() function returns the current time with a high precision. To show an inverted image every 4 seconds, an if condition is included which tests whether the difference between the start time and current time is divisible by 4. This difference is passed through an int() function which truncates the decimal part. Hence, every time difference from 4 to 4.9 is treated as 4. This allows the inverted image to be shown for exactly 1 second, which is expected.

**Problems**:

For the 3rd subtask, I used a rather long and unnecessary approach where I used 2 separate while loops for displaying upright and inverted frames. Even then, due to many loops present, the video feed was laggy.

**Solution**:

To slow the process down, a waitKey() function is added. This delays the execution of the process by a tiny amount not noticable to us, but enough for the process to show the frames as well as flip the frame whenever required

cap = cv2.VideoCapture(0)

while True:

start = time.time()

while time.time()-start<=4:

ret, frame = cap.read()

cv2.imshow('frame', frame)

cv2.waitKey(1)

start = time.time()

while time.time()-start<=1:

ret, frame = cap.read()

flip = cv2.flip(frame, 0)

cv2.imshow('frame', flip)

cv2.waitKey(1)

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

After this, a more efficient and simple approach was used which is shown in the code snippet in the beginning.