Practice Project - OpenCV based : People Counting

Table of Contents

In [ ]:

```
import cv2
import numpy as np

# Initialize video capture device
cap = cv2.VideoCapture('data/people-capture.mp4')

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))

# Load the trained Haar cascade classifier for detecting people
person_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')

# Initialize variables for counting people
count = 0
is_counting = False

while True:
    # Read a frame from the video capture device
    ret, frame = cap.read()
    if not ret:
        break

    # Convert the frame to grayscale for detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect people in the grayscale frame
    people = person_cascade.detectMultiScale(gray, 1.1, 4)
```

```
    # Draw rectangles around the detected people
    for (x,y,w,h) in people:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

    # Check if people are entering
    if len(people) > 0 and not is_counting:
        count += 1
        is_counting = True
    elif len(people) == 0 and is_counting:
        is_counting = False

    # Display the frame with the detected people and the count
    cv2.putText(frame, "Count: {}".format(count), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
    cv2.imshow('frame',frame)

    # Write the frame to the output video file
    out.write(frame)

    # Wait for key press and check for 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release everything and close the windows
cap.release()
out.release()
cv2.destroyAllWindows()
```

In [ ]:

```
import cv2
import numpy as np

# Initialize video capture device
# cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture("data/people-capture.mp4")
xvalues = list()

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('data/output.avi',fourcc, 20.0, (640,480))

# Load the trained Haar cascade classifier for detecting people
person_cascade = cv2.CascadeClassifier('haarcascade_fullbody.xml')

# Initialize variables for counting people
```

```
count = 0
is_counting = False

while True:
    # Read a frame from the video capture device
    ret, frame = cap.read()

    # Convert the frame to grayscale for detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect people in the grayscale frame
    people = person_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw rectangles around the detected people
    for (x,y,w,h) in people:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

    # Check if people are entering
    if len(people) > 0 and not is_counting:
        count += 1
        is_counting = True
    elif len(people) == 0 and is_counting:
        is_counting = False

    # Display the frame with the detected people and the count
    cv2.putText(frame, "Count: {}".format(count), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv2.LINE_AA)
    cv2.imshow('frame',frame)

    # Write the frame to the output video file
    out.write(frame)

    # Wait for key press and check for 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release everything and close the windows
cap.release()
out.release()
cv2.destroyAllWindows()
```

Problem Statement:

```
Fancy indexing allows to select entire rows or columns out of order on a numpy array.
```

1. Fancy Indexing

Fancy indexing allows you to select entire rows or columns out of order,to show this, let's quickly build out a numpy array:

2. Importing Libraries

In [ ]:
```
import numpy as np
```

3. Python implementation

In [ ]:
```
#Set up matrix
arr2d = np.zeros((10,10))
```

In [ ]:
```
#Length of array
arr_length = arr2d.shape[1]
print("Array Length", arr_length)
```

```
Array Length 10
```

In [ ]:
```
#Set up array

for i in range(arr_length):
    arr2d[i] = i

arr2d
```

Out[ ]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5., 5., 5., 5., 5.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.],
       [9., 9., 9., 9., 9., 9., 9., 9., 9., 9.]])
```

Fancy indexing allows the following

In [ ]:
```
arr2d[[2,4,6,8]]
```

Out[ ]:
```
array([[2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.]])
```

In [ ]:
```
#Allows in any order
arr2d[[6,4,2,7]]
```

Out[ ]:
```
array([[6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.]])
```
