Using Python to Track an Object in Video

Subject : IPMV

Div: TE-A | EXTC

Group No : 9

Executive Summary :-

Tracking a Object in a video is an important taks in industries as well as day to day life to track traffic, automated manufacturing inspection, and other automation fields. This Documentaion shows how a red ball in the video is tracked through a maze and show its coordinates at each frames.

Introduction :-

Python is a high level programming language. Being interpreted it is slower than other low level languages It supports many modules that overcome this deficiency and also many computer vision and image processing modules.

Requirements :-

OS: Windows/Linux.

Software: Python 3.5+ with cv2,numpy and os modules installed.

Python Script:

https://github.com/JiteshKanojia/IPMV-mini-project

Process :-

First we start by importing the module we need in python that we will further require

1. OpenCV -> Open Source image Processing library. Highly Optimized for realtime applications.

2. Numpy -> Provides Multidimensional arrays in Python with High level mathematical functions to apply to these arrays.

3. OS -> Provides Functions and methods for the program to interact with the Operating System.

```
1
2
3  # Author :              [Jitesh Kanojia]
4  # Filename:             task_1a_part1.py
5  # Functions:            process_video
6  #  Global variables:    frame_details
7
8
9  ######################## IMPORT MODULES ########################
10 ## You are not allowed to make any changes in this section. ##
11 ## You have to implement this task with the three available ##
12 ## modules for this task (numpy, opencv, os)                ##
13 ################################################################
14 import cv2
15 import numpy as np
16 import os
17 ################################################################
18
```

The import Keyword is used to Import the modules
cv2 , numpy , os

The Main Function consists of all the Operating System related code that deals with console messages as well as file handling and error excpetions handling.

```
90
91  # Function Name:    main
92  #        Inputs:    None
93  #       Outputs:    None
94  #       Purpose:    the function first takes input for selecting one of two videos available in Videos folder
95  #                   and a input list of frame numbers for which the details are to be calculated. It runs process_video
96  #                   function on these two inputs as argument.
97  if __name__ == '__main__':
98
99      curr_dir_path = os.getcwd()
100     print('Currently working in ' + curr_dir_path)
101
102     # path directory of videos in 'Videos' folder
103     vid_dir_path = curr_dir_path + '/Videos/'
104
105     try:
106         file_count = len(os.listdir(vid_dir_path))
107
108     except Exception:
109         print('\n[ERROR] "Videos" folder is not found in current directory.')
110         exit()
111
112     print('\n-----------------------------------------')
113     print('\nSelect the video to process from the options given below:')
114     print('\nFor processing ballmotion.m4v from Videos folder, enter \t=> 1')
115     print('\nFor processing ballmotionwhite.m4v from Videos folder, enter \t=> 2')
116
117     choice = input('\n==> "1" or "2": ')
118
119     if choice == '1':
120         vid_name = 'ballmotion.m4v'
121         vid_file_path = vid_dir_path + vid_name
122         print('\n\tSelected video is: ballmotion.m4v')
123
124     elif choice == '2':
125         vid_name = 'ballmotionwhite.m4v'
126         vid_file_path = vid_dir_path + vid_name
127         print('\n\tSelected video is: ballmotionwhite.m4v')
128
129     else:
130         print('\n[ERROR] You did not select from available options!')
131         exit()
132
133     print('\n-----------------------------------------')
134
135     if os.path.exists(vid_file_path):
136         print('\nFound ' + vid_name)
137
138     else:
139         print('\n[ERROR] ' + vid_name +
140               ' file is not found. Make sure "Videos" folders has the selected file.')
141         exit()
142
143     print('\n-----------------------------------------')
144
145     print('\nEnter list of frame(s) you want to process, (between 1 and 404) (without space & separated by comma) (for example: 33,44,95)')
146
147     frame_list = input('\nEnter list ==> ')
148     frame_list = list(frame_list.split(','))
149
150     try:
151         for i in range(len(frame_list)):
152             frame_list[i] = int(frame_list[i])
153         print('\n\tSelected frame(s) is/are: ', frame_list)
154
155     except Exception:
156         print('\n[ERROR] Enter list of frame(s) correctly')
157         exit()
158
159     print('\n-----------------------------------------')
160
161     try:
162         print('\nRunning process_video function on', vid_name,
163               'for frame following frame(s):', frame_list)
164         frame_details = process_video(vid_file_path, frame_list)
165
166         if type(frame_details) is dict:
167             print(frame_details)
168             print('\nOutput generated. Please verify')
169
170         else:
171             print('\n[ERROR] process_video function returned a ' +
172                   str(type(frame_details)) + ' instead of a dictionary.\n')
173             exit()
174
175     except Exception:
176         print(
177             '\n[ERROR] process_video function is throwing an error. Please debug process_video function')
178         exit()
179
180     print('\n-----------------------------------------')
181
```

The Main Function calls the function *process_video* with takes two parameters :-

1. **vid_file_path** - This is the file path to the videos which has the object that needs to be tracked.
2. **frame_list** - This is a list that has the frames at which the object is tracked as entered by the user.

The process_video function does the image processing required to get the co-ordinates of the object.

```
cap = cv2.VideoCapture(vid_file_path)

for frameValue in frame_list:
    lower = np.array([200, 20, 9])
    upper = np.array([225, 45, 13])

    flagg = cap.set(cv2.CAP_PROP_POS_FRAMES, frameValue)
    _, frame = cap.read()

    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    filtered_img = cv2.medianBlur(rgb_frame, 3)

    mask = cv2.inRange(filtered_img, lower, upper)
    ret, thresh = cv2.threshold(mask, 127, 255, 1)
    contours, hierachy = cv2.findContours(
        thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        M = cv2.moments(contour)
        if(M['m00'] == 0):
            continue

        else:
            cx = int(M['m10']/M['m00'])

            cy = int(M['m01']/M['m00'])
            # print(cx)
            # print(cy)
            frame_details[frameValue] = [cx, cy]
# print(M)

#####################################################

return frame_details
```

1. First we get the capture variable which has the video.
2. Then we Iterate over the frame_list variable so that we can process each frame in the list.
3. Then we create a Numpy Array having the upper and lower threshold values which we later use for creating the mask.
4. Then we select the frame which we will use using the set method inside the capture variable and use the read method to read the set frames values in BGR [Blue-Green-Red] format (OpenCV default uses BGR format).
5. We then create a RGB image using the cv2.cvtColor method this image still needs to be filtered for better results.
6. We use MedianBlur [3 x 3] to filter the image using cv2.medianBlur() method.
7. Then we create a mask using the cv2.inRange() method which takes the filtered image , lower limit , upper limit as parameters and ouputs a binary image.
8. We then use cv2.threshold() function to get the object which we get in the mask image.
9. Then we use the cv2.findContours() function to find the contours in the image and iterate over the countour to find the moments using the cv2.moments() method which returns a dictionary.A moment is the particular weighted average of image pixel intesities.
10. If the moment “m00” is 0 we just skip the current contour.If not we can calculate centroid co-ordinates by using the formula.

The centroid is given by the formula:-

$$C_x = \frac{M_{10}}{M_{00}}$$

$$C_y = \frac{M_{01}}{M_{00}}$$

C_x is the x coordinate and C_y is the y coordinate of the centroid and M denotes the Moment.

11. This co-ordinates are stored in a dictionary with the key being the frame value for which the co-ordinates were calulated.

Output :-

```
PS C:\Users\Jitesh\Documents\CPP\Code\Python\Eyantra\task_1a_explore_opencv\task_1a_explore_opencv\Task_1A_Part2> python.exe .\task_1a_part2_edited.py
Currently working in C:\Users\Jitesh\Documents\CPP\Code\Python\Eyantra\task_1a_explore_opencv\task_1a_explore_opencv\Task_1A_Part2

═══════════════════════════════════════════

Select the video to process from the options given below:

For processing ballmotion.m4v from Videos folder, enter         ⇒ 1

For processing ballmotionwhite.m4v from Videos folder, enter    ⇒ 2

⟹ "1" or "2": 2

        Selected video is: ballmotionwhite.m4v

═══════════════════════════════════════════

Found ballmotionwhite.m4v

═══════════════════════════════════════════

Enter list of frame(s) you want to process, (between 1 and 404) (without space & separated by comma) (for example: 33,44,95)

Enter list ⟹ 1,10,20,30,40,50,60,70,80,90,100

        Selected frame(s) is/are:  [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

═══════════════════════════════════════════

Running process_video function on ballmotionwhite.m4v for frame following frame(s): [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
{1: [168, 163], 10: [168, 163], 20: [168, 163], 30: [168, 118], 40: [226, 93], 50: [481, 109], 60: [727, 85], 70: [869, 184], 80: [862, 165], 90: [862, 227], 100: [872, 313]}

Output generated. Please verify

═══════════════════════════════════════════
PS C:\Users\Jitesh\Documents\CPP\Code\Python\Eyantra\task_1a_explore_opencv\task_1a_explore_opencv\Task_1A_Part2>
```

Refrences:-

https://numpy.org/doc/

https://opencv-python-tutroals.readthedocs.io/en/latest/index.html

https://docs.python.org/3/library/os.html