

An Object Tracking for Studio Cameras by OpenCV-based Python Program

Sang Gu Lee, Gi Bum Song, Yong Jun Yang

Department of Computer Engineering,

Hannam University

133 Ojung-dong, Daeduk-gu, Daejon

KOREA

sglee@hnu.kr

Abstract: - In this paper, we present an automatic image object tracking system for Studio cameras on the stage. For object tracking, we use the OpenCV-based Python program using PC, Raspberry Pi 3 and mobile devices. There are many methods of image object tracking such as mean-shift, CAMshift (Continuously Adaptive Mean shift), background modelling using GMM(Gaussian mixture model), template based detection using SURF(Speeded up robust features), CMT(Consensus-based Matching and Tracking) and TLD methods. CAMshift algorithm is very efficient for real-time tracking because of its fast and robust performance. However, in this paper, we implement an image object tracking system for studio cameras based CMT algorithm. This is an optimal image tracking method because of combination of static and adaptive correspondences. The proposed system can be applied to an effective and robust image tracking system for continuous object tracking on the stage in real time.

Key-Words: - Object tracking, CMT algorithm, Studio camera, OpenCV-based Python

1 Introduction

The area of object tracking in the digital image processing has been an attractive research subject and has many applications on the field of computer vision. There are many methods of image object tracking such as mean-shift, CAMshift (Continuously Adaptive Mean shift), background modelling using GMM (Gaussian mixture model), template-based detection using SURF (Speeded up robust features), CMT (Consensus-based Matching and Tracking) [1, 2] and TLD methods. CAMshift algorithm is a modified version of the mean shift algorithm. CAMshift tracking adapts the size of the tracking window to the object using the knowledge of the aspect-ratio of the desired object and the 0-th moment of the kernel. However, in this paper, we implement an image object tracking system for studio cameras based CMT algorithm. This is an optimal image tracking method because of using combination of static and adaptive correspondences. For this, we use OpenCV based Python program. Studio cameras can control the pan, tilt and zoom

operations of the camera lens through a computer system. PTZ cameras as studio camera have the ability to moving right, left, up, down and even zoom. Generally, PTZ cameras cover very large areas compared with fixed cameras which would not be able to do. Therefore, recently PTZ cameras have been used in many applications for security operations of moving object tracking through manually operating or automatic control. In this paper, the main idea is that moving objects as chosen by director on the stage are positioned nearly in the middle part of the monitor screen.

The organization of this paper is as followings. Section 2 tackles CMT algorithm for image tracking. In section 3, the proposed system is discussed. In section 4, experimental results are discussed. And finally, section 5 draws a conclusion.

2 CMT algorithm

CMT (Consensus-based Matching and Tracking) algorithm is to search the key points

in the concerned objects and to track the corresponding the key-points each frame in video.

CMT tracking algorithm is as follows. [1, 2]

CMT Algorithm

- Set initial frame box to extract reference descriptors
(by mouse driven dragging for bound box or by touching face box using Adaboost face recognition algorithm)
- to obtain static correspondences, reference descriptors are matched to candidate descriptors
- to obtain adaptive correspondences, compute optical flow algorithm
- compute partitioning of the correspondences into inliers and outliers
- eliminate the ambiguous reference descriptors
- compute rotate frame box
- continue on to track

In the CMT algorithm, association between candidate descriptors and reference descriptors is performed. Using OpenCV library, we compute the operations on part detection, computing descriptors, operations of optical flow and so on.

3 Proposed System

In the proposed system, object images are captured from studio camera on the stage. These images are transferred to PC via image grabber board, and CMT algorithm extracts moving objects and eliminates the back-ground image. Then, the pre-processing procedures such as filtering and morphological computations (erosion, dilation, open and close operations) are performed. Python program sends the packets such as pan and tilt data to the studio camera in mobile environment. In the monitor, the moving objects are displayed in the middle part of the window screen.

Fig. 1 shows the entire hardware structure in this system.

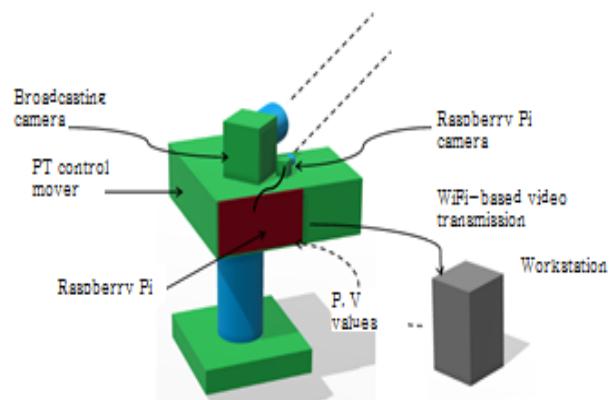


Fig. 1 System structure

In this paper, we use Raspberry Pi 3 Model B and Raspberry camera V.2 to implement object tracking experiments. NOOBS 2.4.3 version is used in Raspbian operation system. We use Python language with OpenCV in the basis of the CMT algorithm.

The overall system architecture of the proposed automatic object tracking system for studio cameras.

First, studio camera is settled on the fixed screw portion of the PT control mover. Inside this, there are 2 stepping motors to execute pan and tilt functions like the actuators.

We experiment the proposed system in the following 2 cases as likes

(1) Raspberry Pi standalone mode

In this, we get real time video image captured in Raspberry Pi camera, execute the OpenCV algorithm and control pan and tilt adjustment in the PT control mover by using GPIO ports in Raspberry Pi.

(2) Wi-Fi transmission and workstation

In this method, real time video image captured in Raspberry Pi camera is transferred to high-

performance workstation in Wi-Fi stream manner. In the workstation, OpenCV based CMT algorithm is executed and the values of the pan and tilt is transferred to the GPIO port in Raspberry Pi in wireless mode to control the PT control mover.

We also basically prepare the lookup table in the first order equation to calculate and translate the moving distance of the PT control mover. The coefficient of the equation for lookup table will be changed more or less to the type and status of main cameras.

```
# look up table
x_weight = 0.29
x_bias = 0.5
y_weight = 0.43
y_bias = 0.8

status, im0 = cap.read()
full_size = im0.shape

x_arr = [i * x_weight + x_bias for i in xrange(full_size[1])]
y_arr = [i * y_weight + y_bias for i in xrange(full_size[0])]
```

Fig. 2 Program code for lookup table

As we can use the lookup table to compute the moving distance easily, the part of the tracking object will be positioned in the centre of the monitor or TV screen.

```
Current center point in frame : (360, 202)
Current center point in recognized box : (301, 163)
To locate the tracked box to center, please move 59 point to left and 39 point t o down

look up table
x value :left 17.61
y value :down 26.17

Current center point in frame : (360, 202)
Current center point in recognized box : (298, 161)
To locate the tracked box to center, please move 62 point to left and 41 point t o down

look up table
x value :left 18.48
y value :down 27.46
```

Fig. 3 Computing the moving coordinates

4 Experimental Results

We use servo motors for PT control mover.

The Python program for setting servo motor of PT control mover is as follows.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
p=GPIO.PWM(17, 50) # 50Hz=20 ms
p.start(7.5)
```

Here we use GPIO 17 and GPIO 18 to servo motor pin as output. Fig. 4 shows the connection of servo motor and Raspberry Pi. J1 and J2 function tilt in the y direction and pan in the x direction pins of servo motor in the PT control mover, respectively.

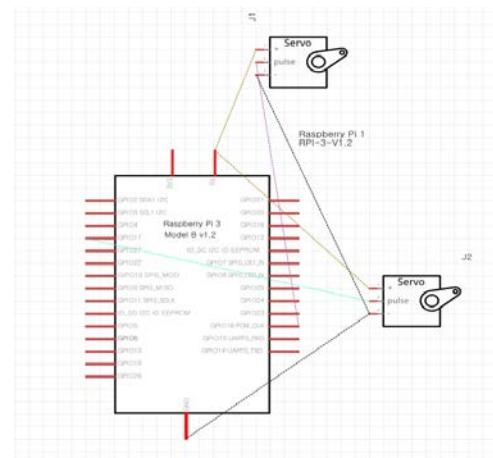


Fig. 4 Connection of servo motor to Raspberry

Servo motors for pan and tilt operation cover near 180 degrees. Using the following ChangeDutyCycle function, for example, servo motor can control the amount of the specific degree.

```
p.ChangeDutyCycle(12.3) # -90 drgree
```

```

p.ChangeDutyCycle(9.0) # -45 drgree
p.ChangeDutyCycle(7.5) # 0 drgree
p.ChangeDutyCycle(5)   # +45 drgree
p.ChangeDutyCycle(2.7) # +90 drgree

```

Control flow is as the followings using CMT algorithm and Raspberry Pi Camera V.2.

```

move_point_x = Centerpoint_in_box[0] -
center_point_in_frame_x
move_point_y = Centerpoint_in_box[1] -
center_point_in_frame_y
Centerpoint_in_box # centroid of tracking object
center_point_in_frame_y # centroid of image view

```

The centroid of tracking object (x, y) can be obtained by CMT algorithm. Therefore, we find the distance between centroid of tracking object and centroid of image view.

```

move_tick_x = move_point_x / 170.0 * -1
if -1.5 < move_tick_x < 1.5 :
    p0.ChangeDutyCycle(move_tick_x + 7.5)

```

```

move_tick_y = move_point_y / 140.0
if -2 < move_tick_y < 2 :
    p1.ChangeDutyCycle(move_tick_y + 7.5)

```

Cover range of the servo motor is between $-90 \sim 90$ degree, and cover range of the values transferred in the ChangeDutyCycle function is $12.3 \sim 2.7$. To shift centroid of tracking object into centroid of image view,

rotating degree of servo motor must be calculated using the moving distance.

Experimental development environment is as followings as Table 1.



Fig. 5 Providing object moving path

Table 1. Development environment

	Development environment		
Hardware	CPU:	Pentium	Quadcore 12.00GHz
	Memory:	8GB	Graphic card : GTX 1050
OS	Microsoft Windows 8.1	Raspbian 4.9	
Software tool	OpenCV 2.4	Python 3	



Fig. 6 Establishment of object retracking-1

As the example of tracking object in the Fig. 5, around of the object to be tracked is displayed in the blue rectangular box. The centre of the blue box is translated into the centre of the yellow circle. We also implement the displaying function of moving path of the object as arrow mark in the Fig. 6.

Fig. 6 and 7 show the situation of re-tracking of new object during tracking current object.

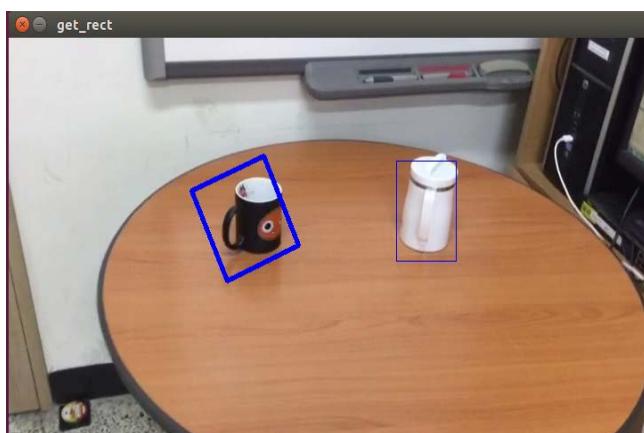


Fig. 7 Establishment of object retracking-2

Changing function of the tracking object is shown in Fig. 6 and Fig. 7.

In order to re-track the new object, if we press “R” key, current frame is stopped and new object frame mark can be created for tracking.

In this paper, methods of taking the object’s frame to be tracked are divided into two ways as following as

(1) Dragging the mouse

To use the mouse in dragging, we can determine the scope of the object to be tracked in the shape of rectangular box. In this method, there are some drawbacks in taking times slightly on setting of new objects and in taking the scope of objects. However, this method is very simple to determine the size and shape of objects.

(2) Select face box

In general, studio cameras move positions according to the faces of the requested people. In studio cameras as DSLR, rectangular boxes of person’s faces automatically shown in the monitor by Adaboost face recognition algorithm.

5 Conclusion

We have implemented an image object tracking system for studio cameras using OpenCV based Python language. This system can control the position of studio cameras in pan and tilt as following the changing of objects in real time. And it can also arrange the moving object images in the middle part of the TV or monitor screen.

For the future study, we will develop a new tracking system that can be utilized in the multi-GPU board in order to speed-up the total performance and to build that system in a stand-alone prototype. As a further research, we will also implement several extensions to the system’s ability for an integrated smart studio camera system being controlled with mobile devices as cellular phone.

Acknowledgement

This work was supported by Innopolis Foundation, granted by the Korea Ministry of Science and ICT.

References:

- [1] Georg Nebehay and Roman Pflugfelder, “Clustering of Static-Adaptive Correspondences for Deformable Object Tracking” IEEE conf. CVPR 2015.
- [2] M. Kristan and et al., “A novel performance evaluation methodology for single-target trackers”, IEEE tr. PAMI 38 (11), 2016.
- [3] Daniel Doyle and et al., “Optical flow background estimation for real-time pan/tilt camera object tracking”, Measurement 48, 2014.
- [4] GitHub, “CMT” <https://github.com/gnebehay/CMT>
- [5] Wiki “OpenCV” <https://ko.wikipedia.org/wiki/OpenCV>

- [6] OpenCV team, "OpenCV"
["http://opencv.org/“](http://opencv.org/)
- [7] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based object tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 5, pp.564–577, May 2003.
- [8] Kari Pulli, Anatoly Baksheev, Kirill Konyakov and Victor Eruhimov, "Real-time computer vision with OpenCV", Communications of the ACM, 2012, 55, 61–69
- [9] S. Huang; and J. Hong; , "Moving object tracking system based on CAMshift and Kalman filter," International Conference on Consumer Electronics, Communications and Networks (CECNet), pp.1423-1426, 2011.