Tutorial on Object Tracking using a Pi-Cam interfaced on a Raspberry-Pi

e-Yantra Team

July 10, 2016

Contents

1 Object Tracking using a Pi-Cam interfaced on a Raspberry-Pi 3

2 Prerequisites 3

3 Hardware Requirement 3

4 Software Requirement 3

5 Theory and Description 3

6 Experiment 4

7 References 17

8 Exercise 18

1 Object Tracking using a Pi-Cam interfaced on a Raspberry-Pi

Objective of this tutorial is to track a object from a given Region-Of-Interest and move the camera according to the object movement using servos.

2 Prerequisites

- Pan-Tilt Camera system interafced R-Pi
- Python and Open CV installed on the R-Pi

3 Hardware Requirement

- Raspberry-Pi B+ Development Board
- Raspberry-Pi Camera Module(Pi Camera)
- Ethernet Cable/Wifi Mdoule
- Camera mount frame

4 Software Requirement

- Python 2.7.11
- OpenCV 2.4.13
- numpy 1.7.1
- RPIO and picamera packages
- **Note :** These are the versions we were working on while creating this tutorial.

5 Theory and Description

Here We will be moving the camera according to the object movement in the frame so that the camera always tries to keep the object at its center.

To make the camera servos move in both Pitch and Roll directions at the same time simultaneously which is not possible as the servos don't have a feedback of how much distance they had moved or the angle they are in present state,and also we cant get a feedback through the center as both the camera and servos need the center at the same time for tracking and movement respectively.

This center controversy can be solved by multi-processing,in which we assign each Roll and Pitch servo a process in which they will be two global variables (one is Current Position of the servo and the second is Desired Position of the servo).These processes will be executing in the back-end all the time,but gets updated only when they get called by a function which updates the two global variables.

For the distance to be moved by the servos as we don't have a feedback from the servo,we can use the proportionality constant which can be obtained by calibrating the servos several times.(Explained in Tutorial 5)
Note: We can also use PID controller for which we need to know the Integration and Derivation constants.

CAMShift tracking is done parallel to this camera movement in which again a center controversy may arise.(Refer to Tutorial 4 for understanding the working of CAM Shift code).Here as we know that the CAM shift works on the previous track window and previous center which will be a totally different co-ordinates when the camera moves frequently.

We can solve this as we know that camera always tries to keep the object at its center.So we gave the CAM shift its previous track window as the camera center.

Here issues may arise about "Is the servo fast enough to catch-up with the CAM shift track by keeping the object at its center every time?"

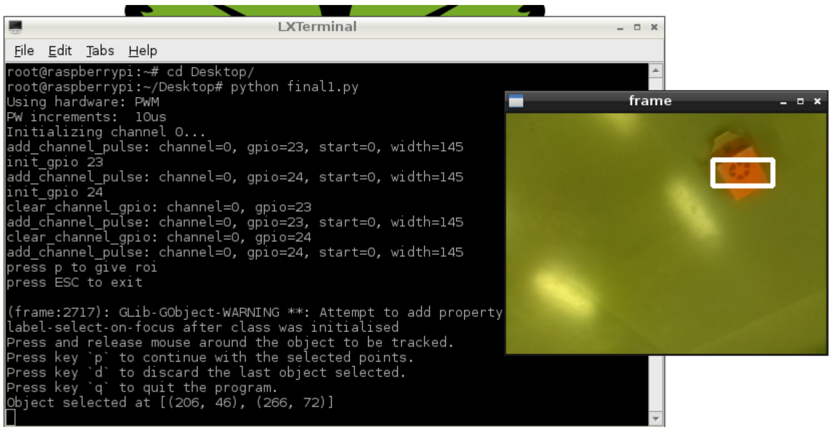
As we know the servos work in a totally different processes ,they are fast enough to meet up our needs of the CAM Shift track window (which we took a 20x20 square from the center).The implementation of this is shown in the below sections.But this solution has not given us optimal performance as the whole system is slowed down.This can be solved by implementing PID controller according to the CAM shift center and camera center movement.

6 Experiment

The Python code is available here

7 Exercise

Object tracking with camera movement is shown below.



```
root@raspberrypi:~# cd Desktop/
root@raspberrypi:~/Desktop# python final1.py
Using hardware: PWM
PW increments:  10us
Initializing channel 0...
add_channel_pulse: channel=0, gpio=23, start=0, width=145
init_gpio 23
add_channel_pulse: channel=0, gpio=24, start=0, width=145
init_gpio 24
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=145
clear_channel_gpio: channel=0, gpio=24
add_channel_pulse: channel=0, gpio=24, start=0, width=145
press p to give roi
press ESC to exit

(frame:2717): GLib-GObject-WARNING **: Attempt to add property
label-select-on-focus after class was initialised
Press and release mouse around the object to be tracked.
Press key `p` to continue with the selected points.
Press key `d` to discard the last object selected.
Press key `q` to quit the program.
Object selected at [(206, 46), (266, 72)]
```

Figure 1: Selction of object through ROI

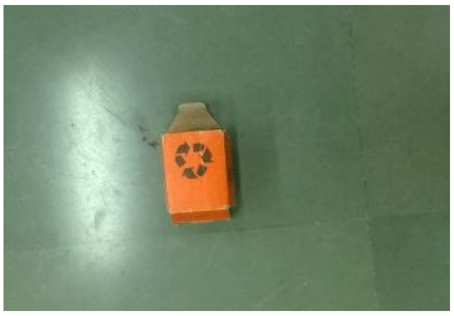


Figure 2: Initial position of the object

8 References

The idea of using multi-processing is taken from ServoBlaster package used in R-Pi for multiple servo controlling written by Richard Hirst.

1. `https://github.com/richardghirst/PiBits/tree/master/ServoBlaster`
2. `https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=54067`
3. `https://www.youtube.com/watch?v=m99n2heDXE87`
4. `http://www.instructables.com/id/Raspberry-Pi-Ball-tracking/`

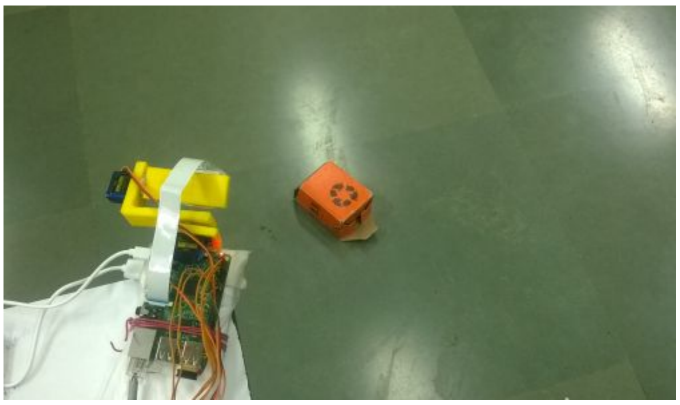
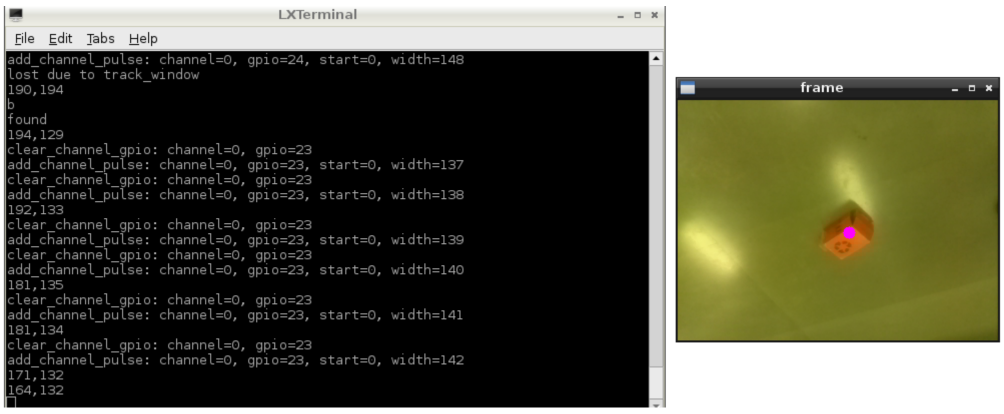


Figure 3: Object slightly moved to left



```
add_channel_pulse: channel=0, gpio=24, start=0, width=148
lost due to track_window
190,194
b
found
194,129
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=137
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=138
192,133
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=139
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=140
181,135
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=141
181,134
clear_channel_gpio: channel=0, gpio=23
add_channel_pulse: channel=0, gpio=23, start=0, width=142
171,132
164,132
```

Figure 4: Camera also slightly moved to left

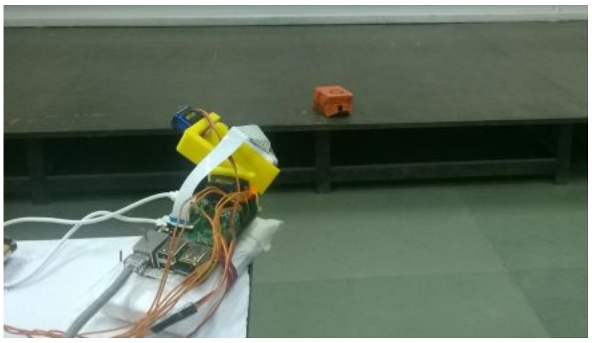
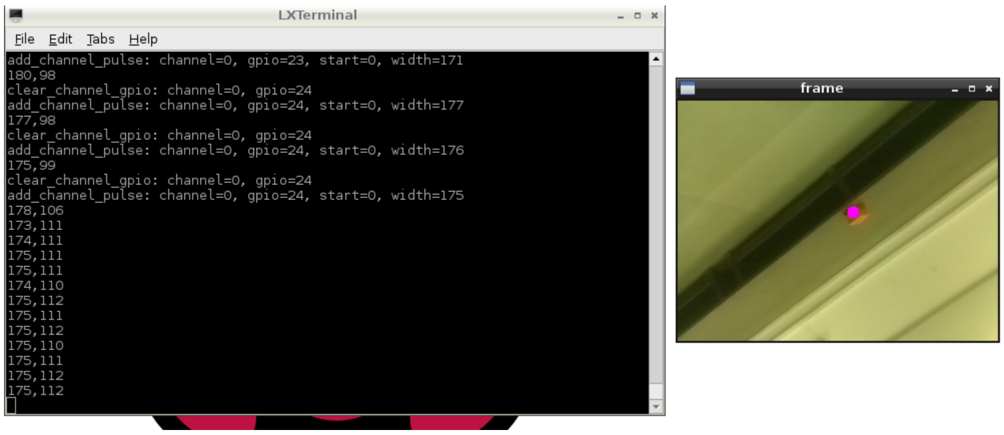


Figure 5: Object largely deviated



```
add_channel_pulse: channel=0, gpio=23, start=0, width=171
180,98
clear_channel_gpio: channel=0, gpio=24
add_channel_pulse: channel=0, gpio=24, start=0, width=177
177,98
clear_channel_gpio: channel=0, gpio=24
add_channel_pulse: channel=0, gpio=24, start=0, width=176
175,99
clear_channel_gpio: channel=0, gpio=24
add_channel_pulse: channel=0, gpio=24, start=0, width=175
178,106
173,111
174,111
175,111
175,111
174,110
175,112
175,111
175,112
175,110
175,111
175,112
175,112
```

Figure 6: Camera sucessfully catching up with the object