

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

e-Yantra Team

July 10, 2016

## Contents

<b>1</b>	<b>Object Tracking using a Pi-Cam interfaced on a Raspberry-Pi</b>	<b>3</b>
<b>2</b>	<b>Prerequisites</b>	<b>3</b>
<b>3</b>	<b>Hardware Requirement</b>	<b>3</b>
<b>4</b>	<b>Software Requirement</b>	<b>3</b>
<b>5</b>	<b>Theory and Description</b>	<b>3</b>
<b>6</b>	<b>Experiment</b>	<b>4</b>
<b>7</b>	<b>Exercise</b>	<b>5</b>
<b>8</b>	<b>References</b>	<b>7</b>

# 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 interfaced 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 Module
- 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 can't 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](#)

The demo can be found [here](#)

## 7 Exercise

Object tracking with camera movement is shown below.

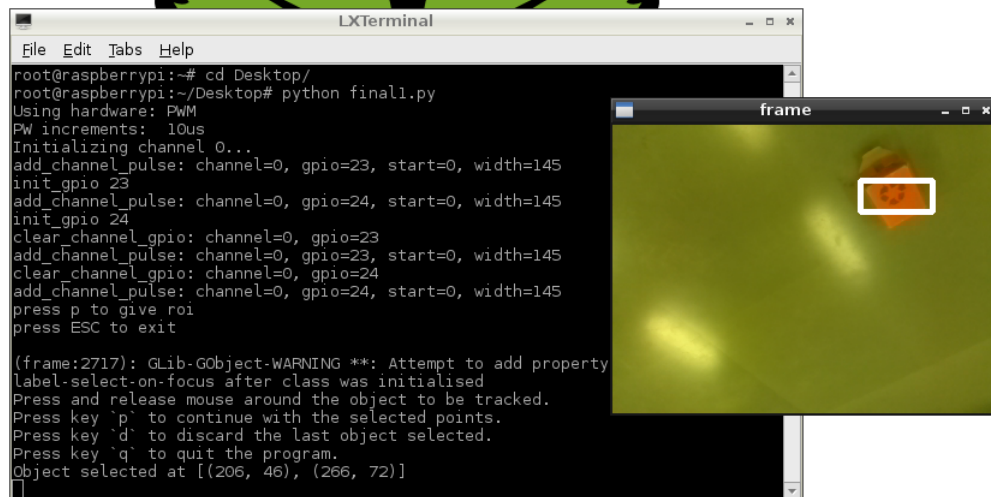


Figure 1: Selection of object through ROI



Figure 2: Initial position of the object



Figure 3: Object slightly moved to left

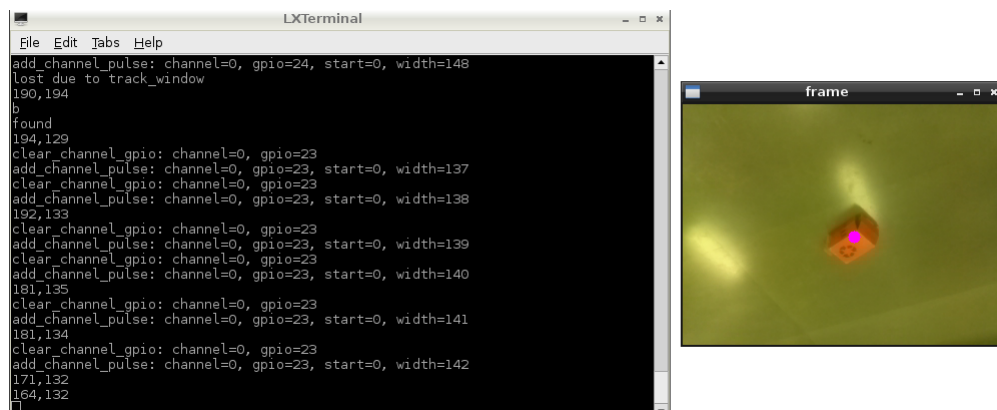


Figure 4: Camera also slightly moved to left

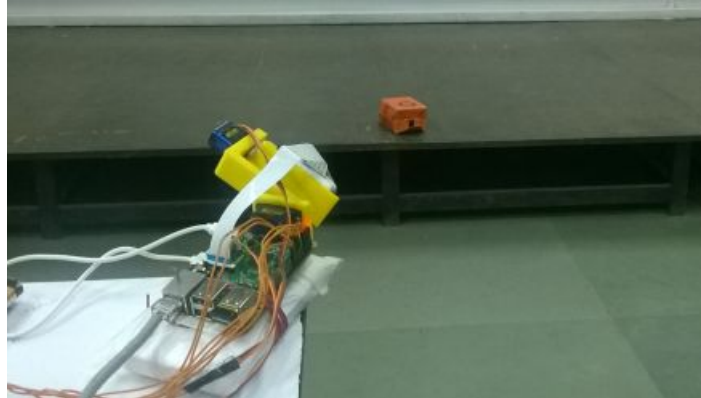


Figure 5: Object largely deviated

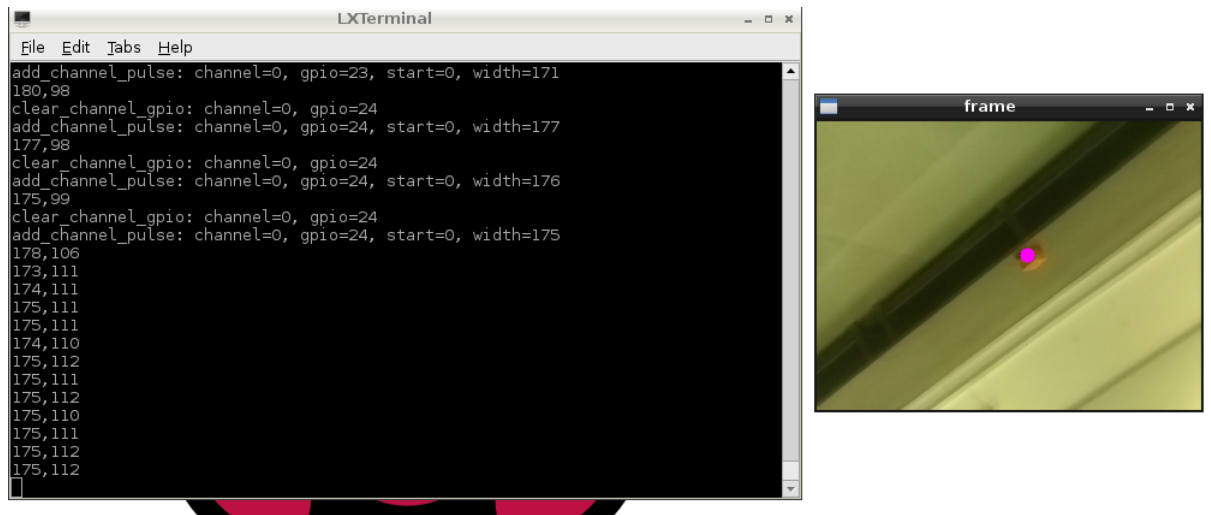


Figure 6: Camera sucessfully catching up with the object

## 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/>