**Object tracking**

### **I. Introduction**

Object tracking project using Udoo Neo board, with core A9 running Linux system which took responsibility for image processing and sent control signal to core M4 to control server motor.

**Project purpose**:

- Select object to detect: Using computer mouse to select object and press ‘c’ button on the keyboard to take template picture of the object and starting detect.

- Sending the object position on the screen to core M4 and starting control server motor to track object.

**II. Analyzing and finding solution.**

**1. Analyzing.**

- Prepare: Udoo board, 2 servo motor, LCD screen and USB camera

- To achieve project purpose, we need to find out the efficient solution of image processing and control smoothly servo motor to track the object.

**2. Finding solution**

In this project the important and difficult part is image processing, because precisely object detecting just send correct control signal. In image processing there are a lot of solution and algorithm support object detecting. The following is some algorithms we found and why we just used final one.

**2.1. *Cascade Classifier Training***

The work with a cascade classifier includes two major stages: training and detection.

For training we need a set of samples. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects. Set of negative samples must be prepared manually, whereas set of positive samples is created using opencv\_createsamples utility.

To detect object efficiently we need thousands of negative images and hundred positive images and take several hours to training and create the xml file. So we cannot apply this method to our project because take so many time to training as well as need so many sample pictures.

**2.2 Motion detection algorithm**

**Motion detection** is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. Motion detection can be achieved by either mechanical or electronic methods. When motion detection is accomplished by natural organisms, it is called motion perception.

A simple algorithm for motion detection by a fixed camera compares the current image with a reference image and simply counts the number of different pixels. Since images will naturally differ due to factors such as varying lighting, camera flicker, pre-processing is useful to reduce the number of false positive alarms.

More complex algorithms are necessary to detect motion when the camera itself is moving, or when the motion of a specific object must be detected in a field containing other movement which can be ignored. An example might be a painting surrounded by visitors in an art gallery. For the case of a moving camera, models based on optical flow are used to distinguish between apparent background motion caused by the camera movement and that of independent objects moving in the scene.

It’s very hard to explain more detail this algorithm. So we can see the picture below:

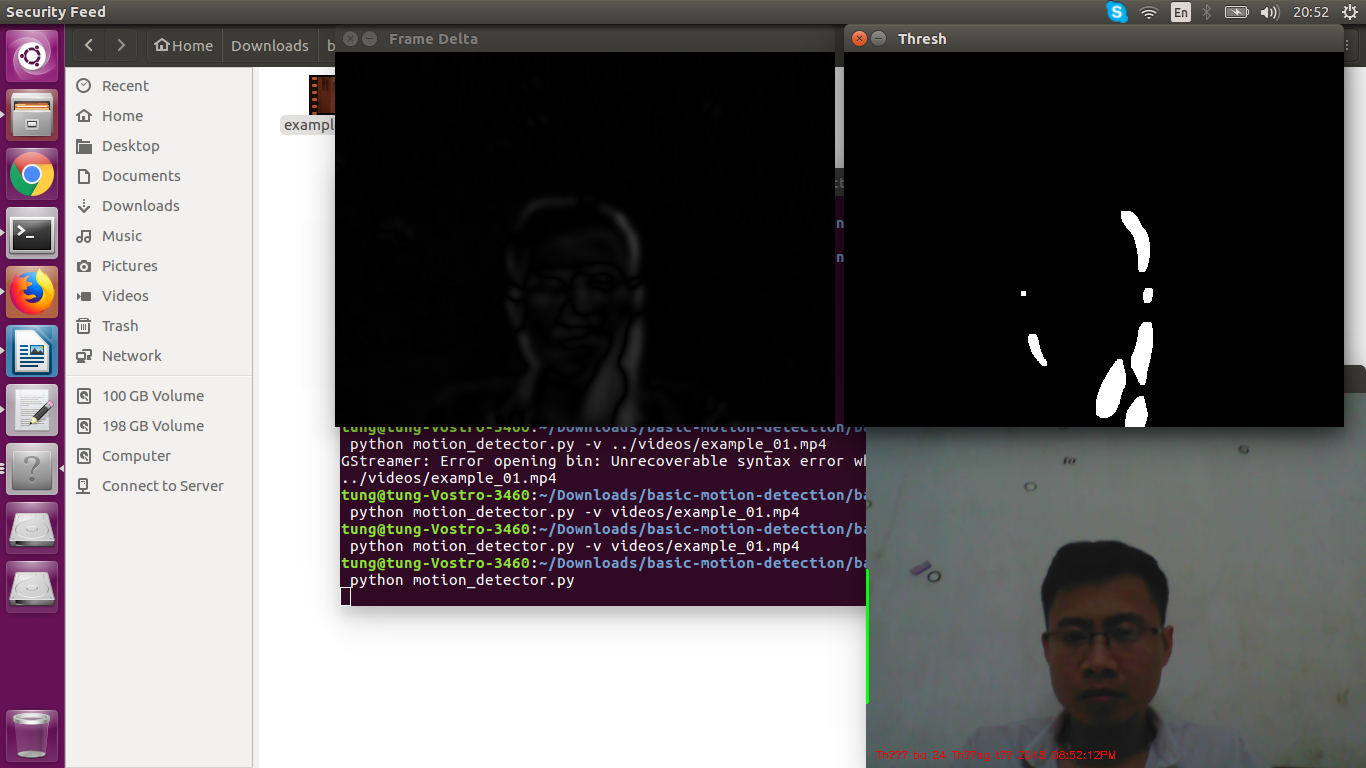


Figure 2.1: Motion detection

Any the motion of object will be highlight with white color on the black ground. So it will based on the white color to detect object.

The **advantage** of this method is detecting correctly object which was moved. The **disadvantage** of this method is object must be moved, cannot detect idle object. So we cannot apply this solution to our project.

**2.3 Color Detection**

Tracking objects based on color is one of the quickest and easiest methods for tracking an object from one image frame to the next.

First, let's start by looking at an image which contains an object to be tracked.



Figure 2.2: Green ball to be detected.

The sample image taken from BucketBot's camera contains a green ball. We'd like to move the robot away from the ball (i.e. backwards) when the ball approaches and move towards the ball (forward) when it moves away.

We use the RGBFilter of RoboRealm to remove all objects in the image except the green ball. The RGBFilter is set to filter out green objects using an RGB channel subtraction and then normalizes the remaining values. This will highlight green objects nicely.

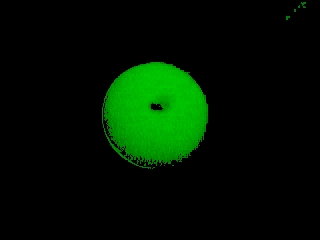


Figure 2.3: Color object to be detected

Now that the ball is segmented from the background we need to determine the objects relative size from image to image to determine if we need to move forward, backwards or remain stationary.

Above is the simple example using color detection. One more other example to let you more understand. See the picture below:



Figure 2.4: Blue color detection

Object with blue partition will be highlight with white color on the black background.

The **advantage** of this method is good for detecting object, but the **disadvantage** is only detecting object with one color that was defined, such as: red, green, blue, yellow… So, it will detect another object with same color, it means that if there are two object with same color it will detect both. So with our project purpose, this method’s not match.

**2.4 Object tracking using Template matching algorithm. [2]**

Template Matching is a method for searching and finding the location of a template image in a larger image. OpenCV comes with a function **[cv.matchTemplate()](https://docs.opencv.org/trunk/df/dfb/group__imgproc__object.html" \l "ga586ebfb0a7fb604b35a23d85391329be)** for this purpose. It simply slides the template image over the input image (as in 2D convolution) and compares the template and patch of input image under the template image. Several comparison methods are implemented in OpenCV. It returns a grayscale image, where each pixel denotes how much does the neighborhood of that pixel match with template.

If input image is of size (WxH) and template image is of size (wxh), output image will have a size of (W-w+1, H-h+1). Once you got the result, you can use **[cv.minMaxLoc()](https://docs.opencv.org/trunk/d2/de8/group__core__array.html" \l "gab473bf2eb6d14ff97e89b355dac20707)** function to find where is the maximum/minimum value. Take it as the top-left corner of rectangle and take (w, h) as width and height of the rectangle. That rectangle is your region of template.

To understand this algorithm, we see the example. We will search for Messi's face in his photo. So I created a template as below:



Figure 2.5: Template image

We will try all the comparison methods so that we can see how their results look like:



Figure 2.6: The result with cv2.TM\_CCOEFF method



Figure 2.7: The result with cv2.TM\_CCOEFF\_NORMED method

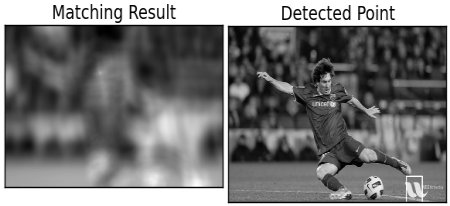
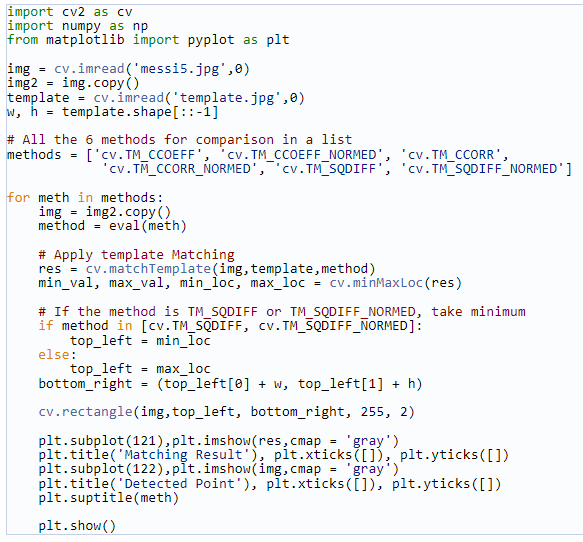


Figure 2.8: The result with cv2.TM\_CCORR method



Figure 2.9: The result with cv2.TM\_SQDIFF method

**The sample code for this solution:**



The **advantage** of template matching algorithm is:

- This is algorithm used much for detecting object.

- Allowing select any template picture which corresponded to the object you want to detect, This matches with project requirement.

- Detecting object relatively efficient in some cases.

The **disadvantage:**

**-** When move object father the detecting less efficiently.

- Can not detect object when object is changed side

**Summary:**

There are a lot of algorithm in detecting object, each one has advantage and disadvantage, based on project purpose to choose the suitable one.

**III. Communication between core A9 and core M4 in Udoo board. [3]**

**1. Internal cores communication**

To communicate between the two cores, Udoo Board has implemented the dedicated Serial object. It is a virtualized serial that uses the shared memory to exchange datas on chip.

When you have a sketch that sends data to the serial device (/dev/ttyMCC) it has to be read by the A9 part otherwise it will slow down the execution: it will time out every Serial.println() function.



Figure 3.1: Communication between core A9 and M4

Sample code with serial port in core M4:

Serial.begin(115200);

Serial.print('Hello');

Serial.print(" ");

Serial.println("A9 core!");

It' possible to access this serial by A9 side on device file: /dev/ttyMCC

**2. External pinout communication**

The iMX 6SoloX can communicate with external peripherals in different modes.

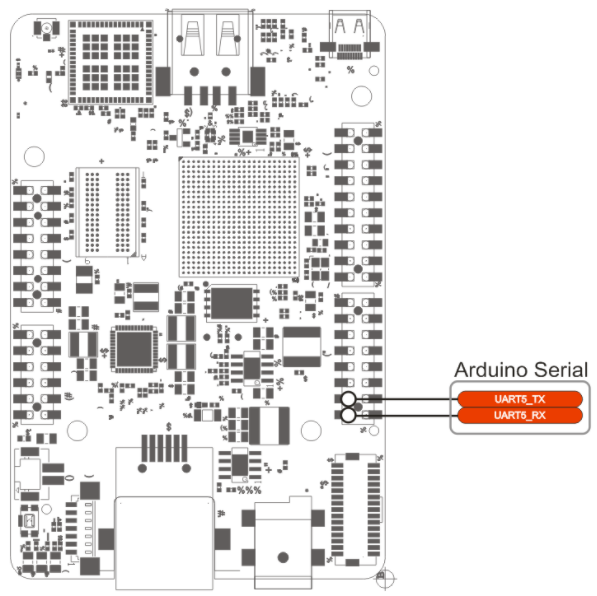


Figure 3.2: External pinout

The Serial0 object allows to read and write data on pins 0 and 1 of external pinout.

This serial is not connected with the A9 core. It allows only to communicate with an external UART device!

**3. Example**

Access the Arduino capabilities of the board in one of the following ways:

* Via an HDMI/LVDS screen or VNC viewer, using *Arduino IDE*
* Access the UDOO Neo desktop;
* Click on Start -> Programming -> Arduino IDE.
* Use the web control panel
* Connect to the board (egg. [192.168.7.2](http://192.167.7.2/) if you are using the USB port);
* Click on *Arduino Editor* on the left
* Use the Arduino IDE installed [on your computer](https://www.udoo.org/docs-neo/Arduino_M4_Processor/Programming_Arduino_M4_from_External_PC.html)

Then, copy and paste the following sketch in the IDE:

|  |
| --- |
| void setup(){  Serial.begin(115200);  Serial0.begin(115200);  pinMode(13, OUTPUT);  }  void loop(){  Serial.print("Hello");  Serial.print(" ");  Serial.println("A9!");  digitalWrite(13, HIGH);  delay(1000);  Serial0.print("Hello");  Serial0.print(" ");  Serial0.println("world!");  digitalWrite(13, LOW);  delay(1000);  } |

Click the Upload button and wait untile the message Done uploading appears on the status bar.

Now, connect to the serial ports, to get the strings.

Open a terminal in UDOO Neo (using the Terminal application on the Desktop, or a [SSH connection](https://www.udoo.org/docs-neo/Basic_Setup/Remote_Terminal_(SSH).html))

minicom -D /dev/ttyMCC

You will see Hello A9!

Reference:

1. <http://www.roborealm.com/tutorial/color_object_tracking_2/slide010.php>
2. <https://docs.opencv.org/trunk/d4/dc6/tutorial_py_template_matching.html>
3. https://www.udoo.org/docs-neo/Arduino\_M4\_Processor/Communication.html