Lab 5

Computer Vision

Author:

Naveen Shamsudhin, Jonas Lussi, Alexandre Mesot, and Prof. Bradley J. Nelson

Institute of Robotics and Intelligent Systems

Date: 2020

Version: 1.3

Summary: In this lab, you will learn to implement basic algorithms in computer vision for robotics. We will use the OpenCV library to detect and track objects by using two different tracking algorithms.

5.1 Background

We perceive the three-dimensional structure of the world around us with ease and can see how vivid the world is by perceiving light rays through our eyes and analyzing information in the brain. Since the 1960s, computer vision researchers have been developing mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery recorded by cameras. Computer vision algorithms are used for event detection, object recognition, object pose estimation, scene reconstruction, video tracking, learning, indexing, motion estimation, image restoration, and more. Various mechatronics systems, such as humanoid robots (ex: HONDA's ASIMO), autonomous cars, smartphones have several onboard cameras to detect, classify, and be able to understand objects within it's field of view, so as to take meaningful action (Figure 5.1).

In this week's lab, you will learn how to track objects and output their coordinates in an image frame. In many robot applications, tracking several objects allows for interacting with the world around.

5.2 Tracking algorithms

In this lab, we will introduce two different object tracking approaches, which are color tracking and edge tracking. With the two tracking algorithms, you will track objects from images or videos provided to you. You will learn that different methods are suitable in different environments, to track different objects and in different light conditions.

5.2.1 Color Tracking

First, you will implement a color tracking algorithm. The program reads an image from a video or a camera and converts it into HSV (hue-saturation-value) color space. Converting an image into HSV color space gives us the advantage of finding a single value for the hue, which is independent of the multiple shades of the color that is

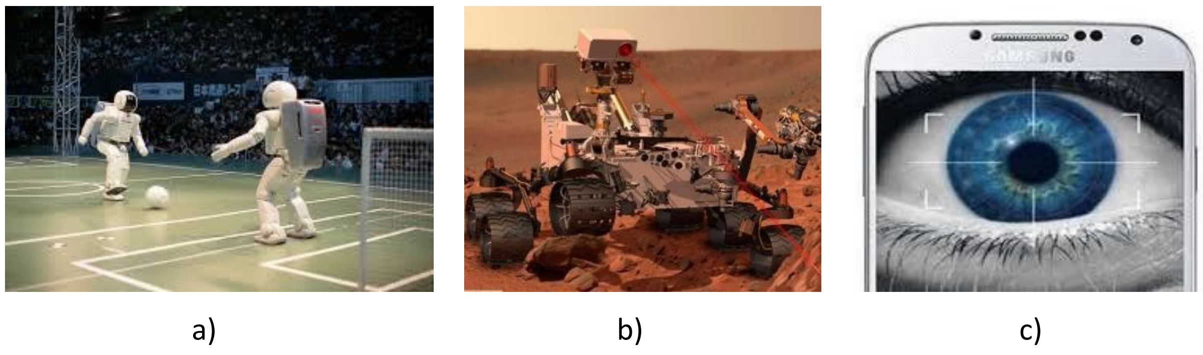

Figure 5.1: Computer vision algorithms find extensive use in (a) humanoid robots, (b) Mars Exploration Rover Mission and (c) Biometric systems in face perception.

being tracked and therefore makes the algorithm less sensible to the surrounding light conditions. Subsequently, a threshold is applied to the HSV image to find a certain color range on the image. The algorithm should find the area of the detected object and derives its centroid. For implementing color tracking on the UDOO, we use OpenCV (see OpenCV Documentation: https://docs.opencv.org/2.4/index.html).

5.2.1.1 HSV Color Space

HSV is the projection of RGB (red, green and blue) color cube onto a chroma angle (Hue), a radial saturation percentage (Saturation), and a luminance-inspired value (Value) as shown in Figure 5.2. In the figure, the value is defined as the height, saturation is defined as a scaled distance from the center axis, and hue is defined as the direction around the color wheel. Another way to interpret the HSV color space is by imagining the mixing of colored paints, where Saturation represents the various tints of brightly colored paint and Value represents the amount by which those paints or mixed with either white paint or black point. Notice that the central axis is completely black at the bottom, and fully white at the top.

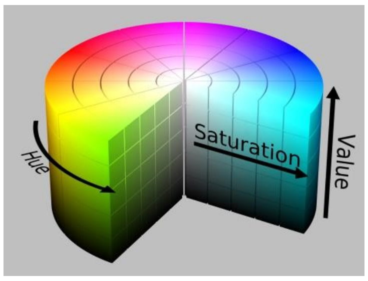

Figure 5.2: HSV color space.

5.2.2 Edge Tracking

The second tracking algorithm you will implement in this lab is edge tracking. Edge tracking is using mathematical methods that are aimed at detecting regions in a digital image that differs in brightness compared to the surrounding areas. The program reads the image from the video, converts it into grey level scale and blurs the image, which smoothens the grey image. Using an edge detection method finds the edges of objects and finally draws the contours of the tracked object (Figure 5.5).

5.2.3 Cameras for Computer Vision

A wide variety of cameras exist on the market for computer vision applications and your choice of camera will depend on many factors (just like any other sensor we use in robotics), such as on the objects that you want to detect, the necessary speed of detection, lighting and temperature of operational environment, form factor of camera, communication protocol between processing system, the total system costs.

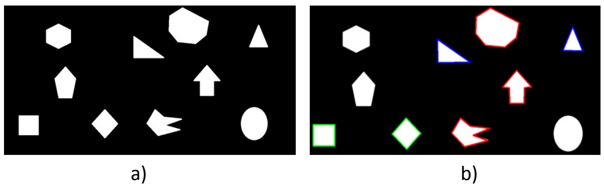

Figure 5.3: Shape detection by using edge tracking method (a) original image (b) tracked image with contour.

5.2.4 OpenCV

OpenCV is an abbreviation of *Open Source Computer Vision* and is library of programming functions for real time computer vision. It has C++, C, Python and Java interfaces running under Windows, Linux, Android and Mac (refer to www.opencv.org).

For the postlab, you will need to include the OpenCV library in your code.

5.3 Prelab Procedure

Note: Prelab assignments must be done before reporting to the lab and must be turned in to the lab instructor at the beginning of your lab session. Additionally, you also have to upload your solution as a single PDF-file to moodle. Make sure to upload the file before your lab session starts, late submissions will not be corrected. The prelab needs to be handed in as a group. All the prelab tasks are marked with **PreLab Qx**.

5.3.1 Color Tracking with Matlab

In the lecture you have seen the basic idea of color tracking. The RGB (red-green-blue) or HSV channels of an image are filtered and a threshold is used to identify areas of interest in the image. In the prelab, you are provided with a single image and a MATLAB skeleton `color_tracker.m`. Write a MATLAB script that reads the image, plots the three hsv channels seperately, uses threshhold on hue and saturation to find the red, green and blue shapes independently and determines the centroid of the object. You can find the details in your skeleton. Your output should look similar to Figure. 5.4. Print and upload your matlab code along with the figures. **(PreLab Q1)**

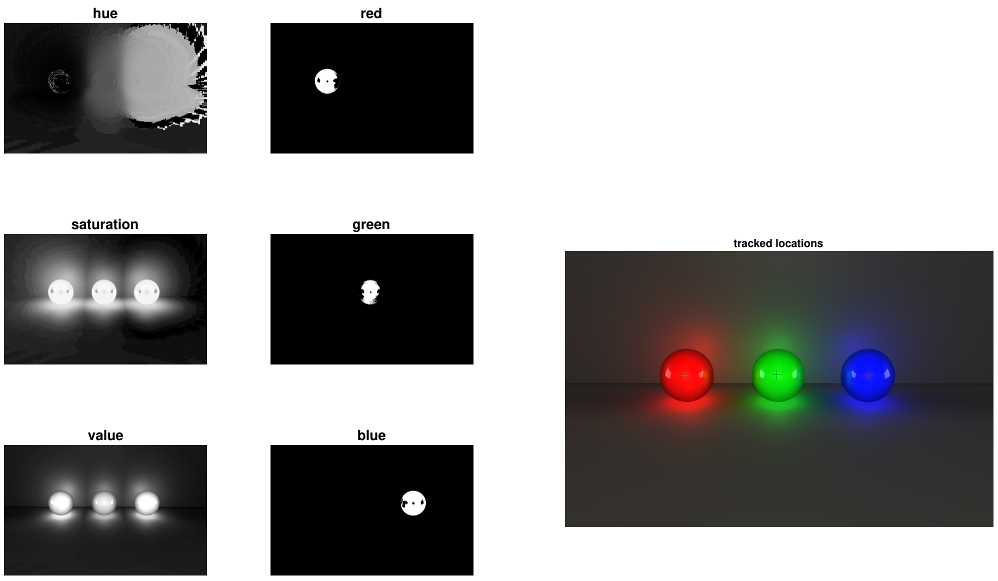

Figure 5.4: Color tracker implemented with MATLAB.

5.3.2 Pixy2 Camera

Pixy2 is a compact and fast camera with an onboard processor built-in. Pixy2 can learn to detect objects that you teach it, just by pressing a button. Pixy2 also has onboard algorithms to detect and track lines for use with line-following robots. Pixy2 does all this at 60 frames-per-second, so your robot can be fast. We will be using the Pixy2 camera for our ball balancing robot which you will model and control in the last lab of the course. For details of the camera, visit their website and for a preview of interesting camera applications for robotics, watch this playlist.

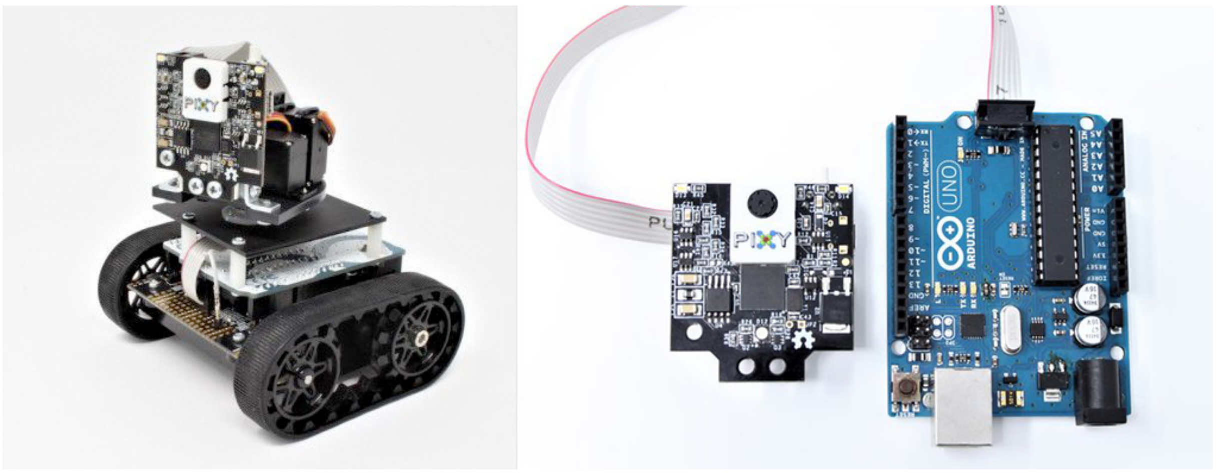

Figure 5.5: Pixy2 camera can be directly connected to compatible Arduinos or to any computer/laptop or Raspbery Pi's with a USB cable.

- What is the primary advantage of using a Pixy2 camera for your robot, as compared to majority of other cameras compatible with Arduino boards (see for example a list of cameras here)? **(PreLab Q2)**

- You have set up the Pixy2 camera to detect green colored balls in its field of view (see 1, 2). The camera detects these green balls and outputs the position of the balls. How can you connect the Pixy2 camera to the microcontroller on the UDOO to read out the position information? What communication protocol can be used to talk between the camera and microcontroller? Draw a schematic of the connections and corresponding pins used on both devices. (hint) **(PreLab Q3)**

5.4 Lab procedure

Note: Questions marked with **Postlab Qx** have to be answered as part of **PostLab 05**.

5.4.1 Color Tracking

1. On moodle you will find a video and skeletons for various color tracking functions.
2. Write a function `ColorTrackingSetColors()` to create color trackbars for tuning the threshold ranges of the HSV image to obtain a visually clear tracked object and save these threshold range. The source code in section **??** is your reference for creating a trackbar with OpenCV. Follow the instructions in the skeleton code. In the main() function, open the provided .avi file and use the `ColorTrackingSetColors()` function you just wrote to set the parameters. Follow the directions in the skeleton code. **(PostLab Q1)**
3. Implement the color tracking function `ColorTracking()` with the help of the comments in the skeleton code. In the main() function, call the `ColorTracking()` function you just wrote on each frame to track the red marker in the .avi file. Use the provided `CrossTarget()` function to target the objects with a cross on the calculated coordinates. Record the trajectory of the tracked objects by saving their coordinates in a txt-file. **(PostLab Q2)**

   You could use the following function to write the data to a text file:

```
// Open .txt file to store all the coordinates
        FILE *coordinate;
        coordinate = fopen("Coordinates.txt", "w+");

        fprintf(coordinate, "%d,%d\t",(int)positionX,(int)positionZ);

        fclose(coordinate);
```

4. Open the txt-file in Matlab and plot the tracked trajectory. Print out and hand in the labelled plot. **(PostLab Q3)**

5.4.2 Edge Tracking

1. Use the `EdgeDetect()` function from your prelab to track the red dot and save the tracked edges in an image. Integrate the `EdgeDetect()` function in the main(). Make sure you are not running two trackers at the same time. **(PostLab Q4)**
2. Try different value of "thresh" (double and half of original value of "edgeparam") to see how results differ with different ranges of Canny edge detection. Save the results and print them. **(PostLab Q5)**

5.5 Postlab and lab report

1. Upload a single PDF-file with your solution to moodle. The file should contain your answers (including plots, images and code) to the postlab questions Q1-Q5 (this includes your complete `tracking.cpp` and `tracking.h` file). Also, make sure to include the plot from PostLab Q3 (Matlab code is not required) and the results and comparison from PostLab Q5. Please upload your solution in time (before next lab session), late submissions will not be corrected.

2. Print the PDF-file and hand it in at the beginning of the next lab session.
3. Show your working code to your assistant in the next lab session. The assistant will ask you to show the following:
   - You should be able to manually select the tracking parameters using ColorTrackingSetColors ()
   - You should be able to demonstrate the Color tracking algorithm by continuously tracking and displaying the red dot (with cross)
   - You should be able to demonstrate the Edge tracking algorithm by continuously displaying the contours of the red dot
4. Come prepared with the Prelab procedure for the next lab.