Hand Tracking Bot

Noel Rohan Vasanth Benjamin Havenaar

May 31, 2021

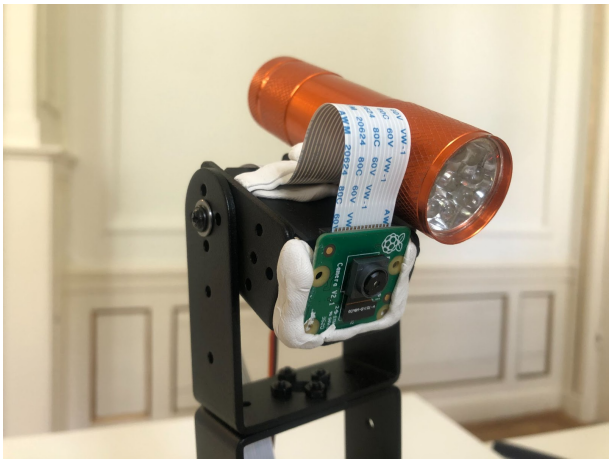

Figure 1: *Hand Track Bot - Camera and light setup*

1 Introduction

In this paper we present the Hand Tracking Bot. A robot arm that recognises and follows a hand and points a light source on it. A practical use for this robot would be to have a light source that follows your hands while performing practical work such as carpentering on a work bench or as a dentist. The project was conducted for the course of Robotics at Leiden University. The entire code for the project as well as a demo video can be found on `https://github.com/noelvasanth/HandTrackBot`.

2 Hardware setup

The Hand Tracking Bot consists of 4 links and three revolute joints (Figure 2). The three servo motors are high torque MG996R motors with a range of 180 degrees. From bottom to top the servo motors are numbered 0 to 2. Motor 1 (in the middle) is always in a fixed position such that link 2 and 3 always have a fixed 90 degree angle. The goal is to make a stand alone robot, therefore we chose to use a Raspberry Pi 3B+ as the controller (Figure 3). On the Pi we use a Servo Driver Hat to control the servos. The Servo Driver HAT has a separate external 6 Volt power supply to power the servo motors. This is external to the power supply given to the Raspberry Pi. At the tip of motor 3 we have attached a Pi Camera and a LED torch light as can be seen from Figure 1.

Robot parts

- Robot arm [1]
- 3x MG996R Servos
- Raspberry Pi 3B+

[1] `https://www.sossolutions.nl/dof-mechanische-robot-arm-met-6-servo-s?gclid=EAIaIQobChMI9IHWsp7b7gIVWeJ3Ch3g_gYiEAQYAiABEgKzefD_BwE`

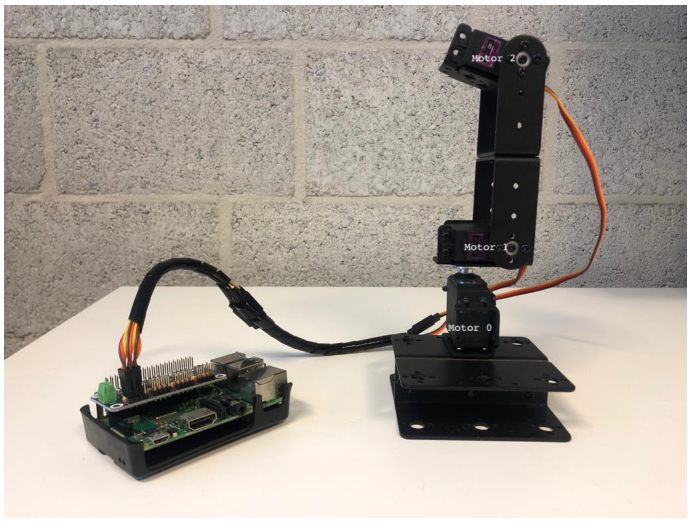

Figure 2: *The robot arm consists of 3 motors connected to the Raspberry Pi through the Servo Driver HAT.*

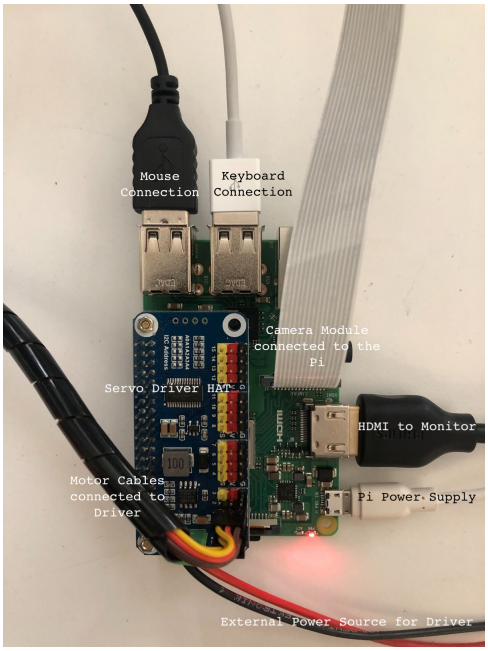

Figure 3: *The connections from the Raspberry Pi to the different components of the setup.*

- Servo Driver Hat [2]
- Pi Camera
- 6 V external power supply
- LED lamp

3 PWM

To control the servo motors we use Pulse Width Modulation (PWM). The frequency of the PWM is set to $50Hz$ with a period of $20ms$. The working is explained with the help of Figure 4.

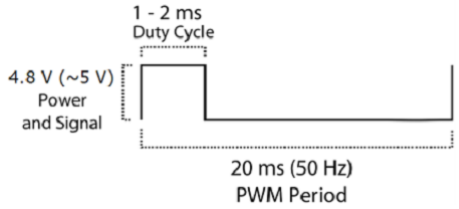

Figure 4: *Duty cycle of the MG996R motors*

Based on the datasheet of the MG996R motor ("MG996R High Torque Metal Gear Dual Ball Bearing Servo", n.d.), the rotation of the motor depends on the width of the pulse given during the duty cycle. If a pulse of width 1ms is given then the rotation is 0°. If the width is till 1.5ms then the rotation will be 90°, and till 2ms the rotation will be 180°. The pulse width was given a value between 500Hz and 2500Hz (2ms - 0.4ms) to control the motors.

Figure 5: *Hand detection*

4 Hand tracking

For hand tracking we use images acquired via the Pi Camera as input source (320 x 240 pixels). We use a Tensorflow Hand tracking model as presented by (Victor, 2017). The authors used a pre-trained model by (Liu et al., 2016) and continued training on the EgoHands dataset (Bambach, Lee, Crandall, & Yu, 2015). The dataset includes 4800 high quality images with pixel level segmentation. The authors used the Tensorflow Object Recognition API with an unspecified CNN architecture to create the hand detection algorithm. To recognize objects, the model splits up the target image in different random bound boxes. The model is then optimized to return a probability of containing a hand per bound box.

After retrieving the probabilities per rectangle we take the rectangle with the highest probability as our bounding box. We choose a probability of at least 0.5 to ensure the robot is idle while no hands are detected. For this implementation we chose to limit the amount of hands to detect to a single one. We then calculate the center of the rectangle and it's distance d in pixels to the camera center. We then calculate angle α_i^t for motor i at time step t as followed:

$$\alpha_i^{t+1} = \alpha_i^t + d_i * k_i$$

Here k_i is a factor, this factor needs to be calibrated separately to translate pixel distance to motor angle for each motor. We found that a factor of $k_{pan} = 3$ and $k_{tilt} = 2$ worked well.

5 Multi threading

Although we used a Tensorflow Light model for the object detection, we still found that the we were only capable of processing 0.9 frames per second. The Raspberry Pi 3B+ has 4 single thread cores built in. We experimented with utilizing more than one core using multi threading. We found that the maximum amount of threads we can allocate to the hand tracking model on the Pi in use was 2. This increased our average frame rates per second to 1.6 frames.

6 Conclusion and further work

We found that it is possible to create a hand tracking bot. Although we were able to implement a multi threaded script to improve performance, we still found that the frames per second where too low to achieve a smooth hand tracking system. With the current implementation the Raspberry Pi 3B+ is not powerful enough to make this system work smoothly. In further work we would like to evaluate other, maybe lighter, implementations of hand tracking. Also we would consider to continue training of the bot with a dataset created specially for the final implementation. Finally we would want to make a hand seeking routine for the moments that the bot does not see any hands in the screen.

[2] `https://www.waveshare.com/servo-driver-hat.htm`

References

Bambach, S., Lee, S., Crandall, D. J., & Yu, C. (2015, December). Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *The ieee international conference on computer vision (iccv).*

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, 21–37. Retrieved from `http://dx.doi.org/10.1007/978-3-319-46448-0_2` doi: 10.1007/978-3-319-46448-0_2

Mg996r high torque metal gear dual ball bearing servo. (n.d.). Retrieved from `https://components101.com/asset/sites/default/files/component_datasheet/MG996R-Datasheet.pdf`

Victor, D. (2017). Handtrack: A library for prototyping real-time hand trackinginterfaces using convolutional neural networks. *GitHub repository.* Retrieved from `https://github.com/victordibia/handtracking/tree/master/docs/handtrack.pdf`