Autonomous Foosball Bot: Man v/s Machine in a game of Foosball

Arkya Chatterjee (Team Leader)

Raunaque Patra

Srijit Dutta

IIT Bombay

Author Note

An ITSP 2016 project under the Robotics Club of IIT Bombay

We, team AutoM8A, acknowledge the mentorship, the technical guidance and the financial support provided to us by the Students' Technical Activities Body (STAB) IIT Bombay, Robotics Club and all other bodies concerned, without whom this project could not have been completed.

Github link to our project: https://github.com/arkyaC/AutoM8A

Abstract

An autonomous foosball bot provides a challenging and entertaining scenario of “man v/s machine” in a game of foosball. It provides an opportunity to the relatively experienced player to play against a formidable opponent without having to search for a human challenger. At present, there are no such machines available in mass production. So our team would like to develop a working prototype of such a device.

Components

- Foosball Table
- Sony PS3 eye camera
- V3006 servo motor x4
- NEMA 17 stepper motors x4
- DRV8825 motor driver for stepper motor
- Belts
- Ramps
- Pulleys
- Arduino Mega
- DC adaptor
- Laptop

Theory Of Implementation

We plan to use image processing to map the entire foosball table on a pair of coordinate axes. The position of the ball at any given time is located using this coordinate mapping. The nearest "player" then tries to kick the ball in the direction of the opponent's goal. Once this functionality is implemented, we plan to streamline the algorithm to develop coordination between the different rows of players.

Mechanical Structure

We used NEMA 17 stepper motors for the linear motion of the rods. They were controlled by DRV8825 drivers. For the angular(rotational) motion of the rods V3006 servo motors were used. For the smooth motion of the rods, the servo motors were mounted on ramps so that it could slide along during the linear motion of the rod. All the motors, pulleys and ramps were placed on an external structure built and placed alongside the table, as shown here:

The hindrance free motion of the servo was ensured by attaching a belt(of a 3-D printer) to it which was connected to the stepper at one end to a pulley fixed at the other end. The belt is kept of sufficient length for the players to traverse any required distance. This ensured that there was no relative movement between the servo motor and the rod it was attached to and thus

the servo could provide the rotational torque whenever required. This arrangement is shown below:

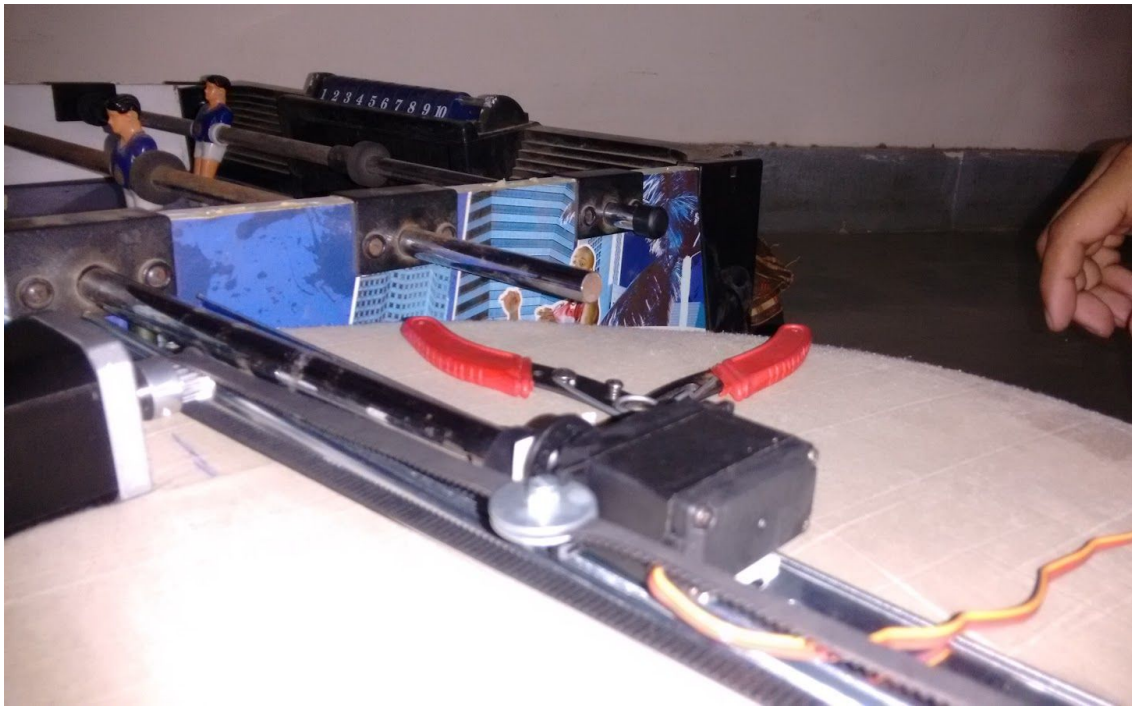


Image Processing

We decided to use ps3 eye camera as it could give upto 120fps (320x240p) and it is relatively cheap (1699 INR). The code is written in python3.4 using OpenCV library as it is quite faster than matlab, which we had considered earlier as another possible alternative.

The state of a foosball game can be modeled by a 2D grid with position data for the players and the balls. The ball is free to move anywhere on this grid so its position can be represented by an x,y coordinate. The players however are limited to one axis, on which they are free to rotate or slide . Furthermore, the players are also fixed in number as well as relative distances between each other. This nature allows to divide the gameState into static and dynamic properties. The static properties are the size of the 2D grid, the number of players per row, maximum and minimum y coordinate of the top most player(one with least y coordinate) in each row, and finally the x-coordinates for each row. The size of each frame was set to 640x480p (60fps). The static properties were determined manually during the start of the program. There are many dynamic properties which include position of each row both

players, angle of each row. However for simplicity we are considering only the coordinates of the ball.

The ball is tracked using a HSV color filter. We choose a green ball as there are no green color elsewhere on the foosball table. The frame captured is passed through the filter after applying a gaussian blur and converting to HSV from RGB. Small objects(noise) in the binary image is removed by applying a series of erode and dilate. The finale binary image contains only the ball.Then contour of the object in the **final** image is found and a best fit circle is drawn. The centre of the circle is the position of the ball. A red dot is drawn on this coordinate for visualisation. The algorithm used was pretty simple and was able to run at a rate of little above 80 fps.

This is a link to our video showing the tracking of the ball by a python code: https://drive.google.com/open?id=0B5g9hHTh9BiTWE5RcGctTVNPUTQ. We managed to track the ball, and then make the goalkeeper follow it as shown in this video: https://drive.google.com/open?id=0B5g9hHTh9BiTRGdteUNpT0hFVzg. Another video showing a partially working goalkeeper rod is shown here: https://drive.google.com/open?id=0B5g9hHTh9BiTZDF0VlBjQUYycU0

Major drawbacks of this algorithm are the HSV filter depends heavily on the surrounding light. If the ball is under a certain rod and only a part of the ball is seen by the camera centre is detected on the side giving a error in position of about (1 cm).

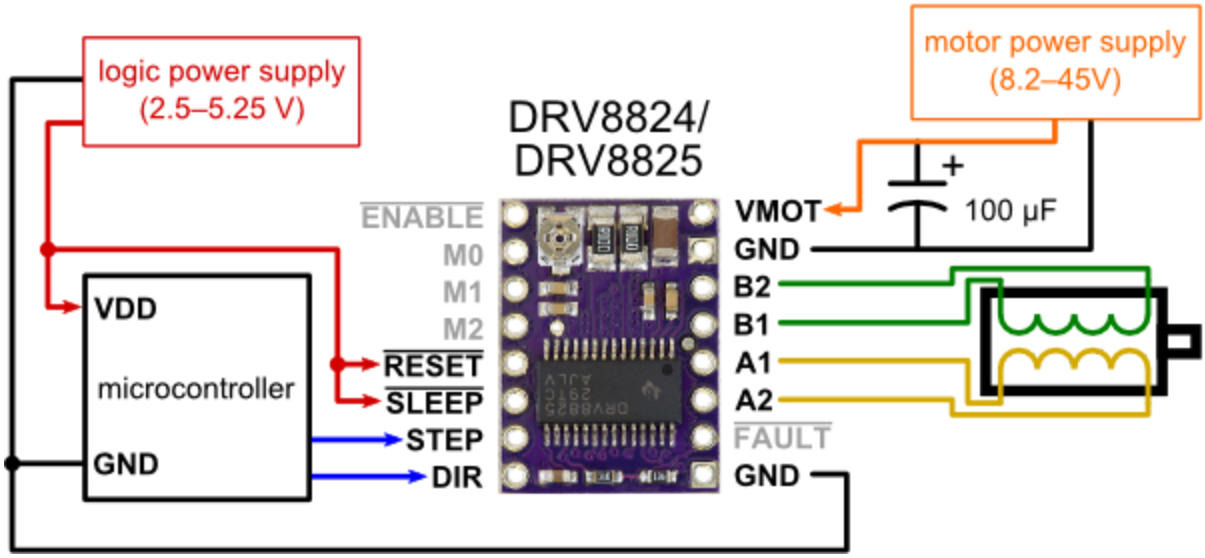
Algorithm To Control Each Motors

Position of the ball (pixel coordinate) is stored in two variables posx and posy and velocity (change in pixel coordinate per frame) is stored in two variables vx and vy. If the ball is in front of a row then a line is drawn in the direction of its velocity and after considering reflection where the line intersects with the row line, that will be the desired position for the foosball men to block the ball. This algorithm assumes the ball to travel in straight lines but it is not so when the velocity is too low, particularly because the foosball table we used was an old one with a surface that was not perfectly flat. Furthermore if the velocity is small the variable vx,vy store small values in the range 0-2,leading to only 7 values for the slope of the line , causing undesirable errors in estimation. Therefore when velocity comes below a certain value or if vx is much lower than vy then the algorithm will switch to a simple algorithm of following the y coordinate of the ball.

Electronics

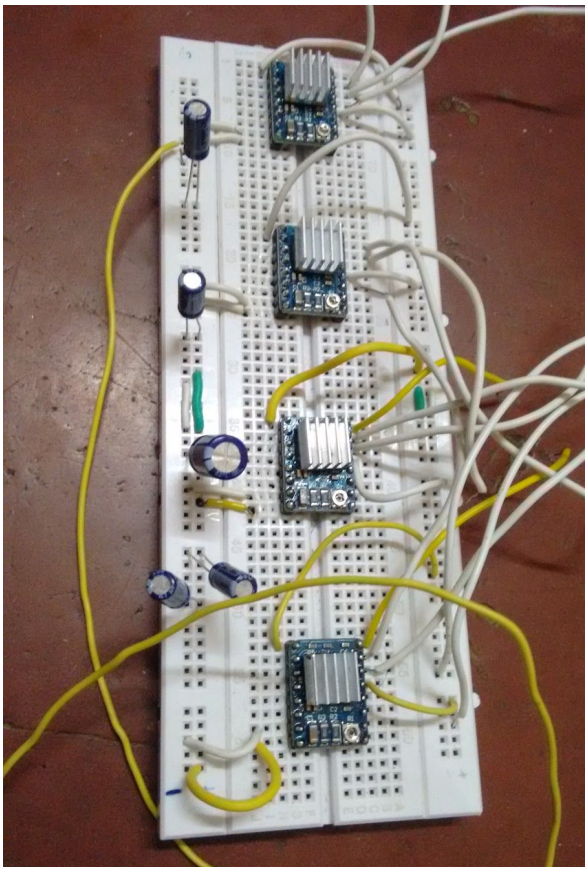
The electronic aspect of our project involved drivers for the stepper motors, which were controlled via an arduino mega. We used DRV8825 motor drivers for controlling the stepper motors. These can handle large amount of current, upto 2.5 amperes, and were our natural choice as our NEMA 17 stepper motors required a maximum current of up to 2 amperes. We used an open source library available on github, which made the use of the drivers extremely simple. In the process we learned a lot about how stepper motors work.

This is the minimal wiring diagram for the DRV8825 driver.



The capacitor used is for decoupling the source from the actual circuit. This protects the driver from experiencing destructive LC voltage spikes, especially when using power leads longer than a few inches. Under the right conditions, these spikes can exceed the 45 V maximum voltage rating for the DRV8825 and permanently damage the board, even when the motor supply voltage

is as low as 12 V. One way to protect the driver from such spikes is to put a large (at least 47 µF) electrolytic capacitor across motor power (VMOT) and ground somewhere close to the board. Here's an image of the driver circuit, without the motor connections:



We used an Arduino Mega for the purpose of motor actuation. The data from the image processing code was sent to the microcontroller via serial communication. We used the pySerial wrapper library for this purpose.

After ball tracking, followed by processing the data to determine which motor should move and by how much, we passed the data in a condensed format via the serial port. This data was used by the Arduino to simply perform the rotation of the motors by the desired amount. In the process of interfacing the Arduino with python code, we realised that the baud rate for sending data would be a major determining factor for the overall latency of the device. However too high a baud rate would tend to increase errors in communication. So we settled on an optimum of 57600. Even then the final structure seemed to lag a bit, which could be probably due to some inefficiency in the algorithm, though this could also have been due to the difference in processing speeds of the laptop and the arduino, or due to the fact that the motors were too slow to catch up with the data stream incoming via the serial connection.

Future Scope

There seems to be tremendous room for improvement and perfection. Considering that this project was completed in a duration of a little over a month, we think we've achieved a significant amount of experience with this problem, more so because most of the other such projects available on the internet have been designed by masters/PhD students at much higher costs, and completed over a much longer period. Having said that, we feel that this project can be taken up from its current state and taken forward to a much better end-product. Making it an embedded system using something like a raspberry-pi or a beaglebone black, would probably make it appear cleaner and more compact. The AI can also be improved upon to a large extent.

References

1. https://www.youtube.com/watch?v=xc7ztgPJkY0#t=47 foosball bot made by students at the Danish Technical University
2. http://www.imagesco.com/articles/picstepper/02.html working of a bipolar stepper motor
3. http://www.electrical4u.com/bipolar-stepper-motor/ working of a bipolar stepper motor
4. https://github.com/laurb9/StepperDriver DRV8825 wrapper library
5. http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/ ball tracking with opencv
6. http://playground.arduino.cc/Interfacing/Python interfacing python with arduino
7. http://aleksandarkrstikj.com/tracking-a-ball-and-rotating-camera-with-opencv-and-arduino/ opencv python and arduino interfacing
8. https://www.youtube.com/watch?v=bSeFrPrqZ2A real-time object tracking with opencv