International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Automated Carrom Bot

Adit Doshi[1], Sushrut Ranade[2], Huzefa Painter[3], Prof. Monika Kanojiya[4]

[1]B.E Student, Dept. of Computer Engineering, SAKEC, Mumbai, Maharashtra, India
[2]B.E Student, Dept. of Computer Engineering, SAKEC, Mumbai, Maharashtra, India
[3]B.E Student, Dept. of Computer Engineering, SAKEC, Mumbai, Maharashtra, India
[4]AssistantProfessor, Dept. of Computer Engineering, SAKEC, Mumbai, Maharashtra, India

---

Abstract - *An automated Carrom bot to play against a human, which can scan the whole board, choose the next shot, calculate angles, and the power required to hit the piece so that it can reach the pocket. After each shot, the Bot can identify the next shot so that it can pot all the coins and win the game. This Bot plays with you and, at the same time, helps you to improve your Carrom skills. It can analyse the coins on the Carrom board using its intelligence and can select the best shot that it can play to win the game. In a world where people prefer outdoor sports over indoor games, people can improve their skills by playing against such a bot, which always has better skills than a human player.*

Index Terms: - Automated Bot, Decision Tree, Object Detection, Carrom, Interdisciplinary.

1. INTRODUCTION:

Pocket Bot is an automated bot that can analyze the coins on the Carrom board, and using its intelligence can select the best shot that it can play to win the game. This Bot is highly accurate, up to 0.01 mm of any movement. This is essential as a slight inaccuracy can result in an unsuccessful shot. Being an amalgamation of multiple motors and sensors, this is a complex build for beginners who have only recently dabbled with robotics. Not only that, the calculations, basic understanding of A.I, and programming knowledge makes this an interdisciplinary project. It cannot only calculate the best shot but also, make sure it is executed perfectly. All this is done solely using the Raspberry Pi 3 including, controlling the motors and all the calculations.

2. LITERATURE REVIEW:

Our focus here is to build a new system by using various predefined algorithms available, combining the algorithms we can simplify the work to be done, such as combining Shape Detection Algorithm, Colour Detection Algorithm to detect and identify various coins present on our carrom board. The first approach here is to make the algorithm work without considering the accuracy of the system, as the accuracy and efficiency of the algorithm can be improved later. Here the initial accuracy of the hardware was close to 0.5mm, but this caused many issues as coins in particular angles couldn't be targeted, so after improving the efficiency and accuracy, we were able to increase the accuracy to 0.1mm which can target almost all coins in all possible angles.

3. PROPOSED SYSTEM:

3.1 Problem Statement:

"To identify the coins present on the board, and increase shot efficiency of the bot to hit each and every possible angle on the board with the best accuracy"

3.2 Proposed System Architecture:

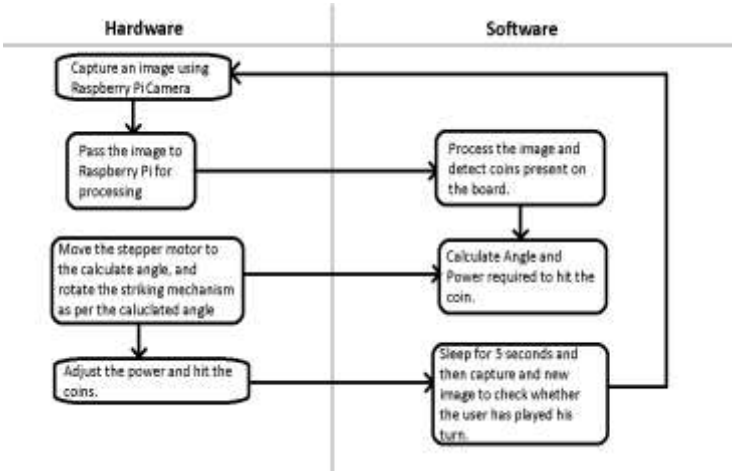

Fig 2.1 : Workflow

3.3 Proposed Methodology:

We will be using Thonny as our base platform for the purpose of coding. Our Methodology involves use of OpenCV, Hough Transform algorithms to detect shapes and colours of our coins so that the bot can process this information and decide which coin is allowed to hit.

Steps involved: -

1. Capture images of the board, and identify various coins present.
2. Process this information to calculate the exact position of the coins.
3. Calculate the power required and then adjust the angle of the striking mechanism.

4. IMPLEMENTATION:

The robot first begins by clicking two images of the board with different positions of the striking mechanism. Combining the two, it looks out for all the coins on the board, and which ones of them are the right color (team) to hit. After this, it calculates which of these has the best chance to be potted in a pocket It then calculates the angle, power, and how much it has to move from its current position to the target position.

Now it commands the Stepper Motor (A NEMA-17) to move a certain number of steps in either direction needed. After this, a set of 4 servo motors come into play (MG 995). A servo first sets the striking mechanism down on the playing board. Then a set of 3 servos on that mechanism work together to: Set the angle of the shot, set the power, and a third servo to lock it in place until released to fire the shot. However, we do not have the resources to implement a striker pick-up mechanism.

Fig 4.1: 3 Servo motors working together and acting as a Striking mechanism to take a shot

Right after it takes the shot, the first servo activates again to lift the mechanism off the playing board. This is done to clear the area for the current, and future shots to play out until it is its own turn again. Horizontal movement is done using the Stepper Motor. The bot rides on a plate situated on a simple screw and nut mechanism, where the stepper rotates the nut and thus moves along the screw.

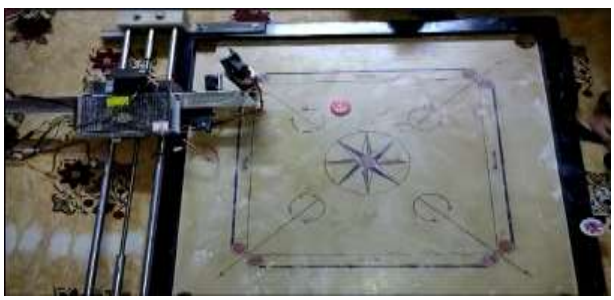

Fig 4.2: Stepper motor carrying bot to its appropriate location

Now, all calculations are done by the Raspberry Pi 3, powered by a power bank situated on the top alongside the camera responsible for taking the images at the start. All the communication between the servo motors is done directly through the GPIO pins. The stepper motor, however, requires a Driver, which the Pi uses GPIO pins to communicate with, and the Driver will, in turn, run the motor accordingly.

Fig 4.3: Raspberry pi 3 situated on top and Pi cam underneath capturing the photo of board

Other calculations in the Raspberry Pi, such as detecting the coins, are done using Hough Circle Transform and Color detection algorithms. The A.I responsible for selecting the shot is self-coded and is a relatively basic decision tree. Future versions will include a Model-Based Agent. This is not done as of writing this paper as previous models for this problem do not exist. However, with a Model-Based Agent, we are sure it will improve the shot accuracy, to a point where professional Carrom players have fierce competition.

Since the image is taken from the same position every time, we have hard-coded the location of the four pockets, as well as the legal bases and baselines. This helps in avoiding trying to detect those lines and circles separately. We first convert the image into grayscale. Then, using image arithmetic, we separate the coins and the other different circles on the board from each other. The grayscale helps in making color detection easier, with the lighter ones being the white team and the darker being black and the middle tone being the queen.

4. A) Working of our Hardware:

Right after identifying the coins, the Pi starts calculating for different shots. It picks the most accessible of them by a measure of how "straight" the line from the striker to the coin to the pocket is. Now for the execution of that shot, we move towards motor calculations. We know all current values for each motor beforehand. It's easy to calculate the distance to move, but we need to convert it to "steps" for the stepper or "degrees" for the servos. For the servos, we need to understand their duty

cycles as Python does not have predefined codes for servos like an Arduino.

"Duty Cycle" is the width of positive pulse (Square wave) or how long it is "active." For a servo with a 2ms Duty Cycle, it means at 2ms Duty Cycle it is at the maximum rotation degree (180 or in some cases 270). At 1ms Duty Cycle, it is at its lowest, i.e., 0 degrees. Now, the servo holds this position with constant updates at the frequency it states. For example, the most common frequency of servos is 50Hz. This means 50 times a second, and it would require a pulse to determine it's next position. So, every 0.02 or 20ms it requires a pulse. Thus, in every 20ms, it needs one Duty Cycle, and therefore it has high accuracy with quick updates. Now, this would be easy with an Arduino where you can set the Duty Cycle directly to be output. However, with a Pi, you cannot do that. Instead, you will need to use software-based PWM (Pulse Width Modulation).

In Python, we have RPi. GPIO library which helps in reprogramming the GPIO pins of the Raspberry Pi. It allows us to use software-based PWM. However, it's never that easy. The PWM needs to be such that it represents the duty cycle, which means mapping a 1-2ms Duty Cycle per 20ms to its PWM equivalent.

This problem is similar to finding a point on a line, where for some value of x, what is the value of y? Where 0,0 might be the start point and 13.7,180 the end. Calculating the slope and thus the formula for this line, we can find any value of y of some x. So now, we can convert any angle required in degrees to a PWM cycle. This is a convenient conversion as we can use this to control all other servos as well—for example, the servo that is responsible for the power. We can decide how much power we need depending on the percentage of rotation between 0-180.

Now for the stepper motor, we have to use a Driver. Again, unlike Arduino, we don't have predefined codes for a stepper too. Again, using RPI.GPIO, we can set up the GPIO pins to simulate how a stepper driver needs input. Depending on how fast it can take steps, we can turn the output on, then off for some delay and back on. This makes the Driver think pulses are coming in as expected. Each pulse means a step, and delays can go down to 0.0001s(In some cases even lower). In this way, speed can also be adjusted depending on how long the delay is. All this is for the STEP pin, which controls how many steps the motor takes. The DIR pin is much simpler, with it having 2 values HIGH or LOW, which can be set for clockwise or anti-clockwise, respectively.

Combining these two pins, we can have any desired movement along the screw. We do have to keep in mind the number of steps it needs to go a complete 360 degrees. This is very important as it can vary across different motors. Each motor might also have a "micro-stepping" mode. This makes the motor take smaller steps to acquire more accuracy. For example, if it takes 200 steps for a complete rotation, a 1/2 stepping mode takes 400 steps for a complete rotation. NEMA-17 has micro-stepping up to 1/32 steps. This makes it very accurate, albeit at a loss of speed.

For the movement to be entirely safe for the motors and equipment, we need to make sure the nut does not exceed the limit of the screw. There are stoppers on both ends of the screw; however, the motor has enough torque to keep spinning the nut over the screw at the end and ruin the threads on it. To avoid this, we keep our initial 0 as the extreme left and the distance from there to the end of the baseline as the right limit. Keeping these hard-coded limits prevents the motor from progressing any further, even if incorrectly commanded to do so.

5. CONCLUSIONS:

The Bot is capable of playing Carrom at a level that is way above average players. However, to beat professionals at this game will require better Artificial Intelligence implementation like the aforementioned Model-Based Agent. The hardware, like the motors and the weight of the equipment, can be improved upon drastically with a higher budget. For example, replacing Aluminum with Carbon Fiber or getting better and stronger Servo and Stepper motors to improve on the speed. We could also use Raspberry Pi plugins to improve the processing power of it to enable implementation of A.I at a higher level, such as live video tracking, to eradicate the need of a button and make it completely autonomous. Adding another axis horizontally along the side can also let us use a grabber to grab the striker automatically.

6. ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude for everyone who has made an invaluable contribution without which the completion of this project would not have been possible. We would like to thank Ms. Monika Kanojiya, Ms. Deepshikha Chaturvedi, and also Mr. Shabbir Painter for their keen interest and meticulous scrutiny and assisting in solving any problems that arose.

REFERENCES

1. Virendra K. Y., Saumya B. , Anuja K. A. , Rahul P. (2014). Approach to accurate circle detection: Circular Hough Transform and Local Maxima concept. Retrieved from IEEE xplore, website: https://ieeexplore.ieee.org/document/6892577

2. Luis Barba-Guamán , Carlos Calderon-Cordova , Pablo Alejandro Quezada-Sarmiento (2017).

Detection of moving objects through color thresholding. Retrieved from IEEE xplore, website: https://ieeexplore.ieee.org/document/7975755

3. Yurong Zhong (2016). The analysis of cases based on decision trees. Retrieved from IEEE xplore, website: https://ieeexplore.ieee.org/document/7883035

BIOGRAPHIES:

Adit Doshi is currently pursuing B.E in Computer Engineering, from Shah and anchor Kutchhi Engineering College, Mumbai. I have completed my Diploma in Computer engineering from K J Somaiya polytechnic Vidyavihar, Mumbai.

Sushrut Ranade is currently pursuing B.E in Computer Engineering, from Shah and anchor Kutchhi Engineering College, Mumbai. He has completed his Diploma in Computer engineering from K J Somaiya polytechnic Vidyavihar, Mumbai.

Huzefa Painter is currently pursuing B.E in Computer Engineering, from Shah and anchor Kutchhi Engineering College, Mumbai. He has completed his Diploma in Computer engineering from Government Polytechnic Mumbai.

Prof. Monika Kanojiya is working as an Assistant Professor in the Computer Engineering Department of SAKEC, Mumbai. She has done her B.E. in Information Technology and M.E in Computer Science. She has 6 years of teaching experience and has guided multiple projects.