Autonomous Checkers Robot

Lilly Chiavetta Austin Camacho Kaelyn Pieter Jared Bailey

Abstract

Utilizing a small robot arm, we present a robotic system capable of autonomously playing a game of checkers against a human opponent. Utilizing its checkers base code and computer vision, the robot will view the human's move, choose its next action, and move the required checker piece appropriately. With four options for the robotic opponent's difficulty, the human player can practice and grow their skills, learning to play checkers against various programs, each with unique strategies for playing. The robotic system includes animatronics and a voice clone of our course TA, in order to resemble the Mechanical Turk of old.

Introduction

The Mechanical Turk was a famous automaton chess player built in the late 1700's, with opponents including Napoleon Bonaparte and Benjamin Franklin. The automaton was operated by a human chess master hidden from sight within its cabinet, providing the illusion that the Mechanical Turk operated autonomously.

As an homage to the Mechanical Turk of old, we set out to build an autonomous and lifelike robot capable of playing a game of checkers against a human opponent.

As the human player acts, they are then able to watch as their robot opponent chooses its best counter move. The robot is capable of removing pieces it jumps, as well as upgrading its pieces to kings.

This project made use of many aspects of current robotic abilities including: ROS 2 simulation, motion planning, pick and place manipulation, computer vision, 3D modeling, animatronics, voice control and cloning, and system integration.

Methods

Animatronics

STL files were obtained from Will Cogley (animatronic eyes) and Fanki the Printer (model of daughters teeth). The printed output from Will's STL files did not fit properly, due to many issues with tolerances. This required multiple adjustments and prints for proper function. The teeth model was not designed for animatronic use, and was modified to allow for the desired movement.

The animatronics were controlled using an arduino, and serial communication for mouth movement to sync with the voice clone.

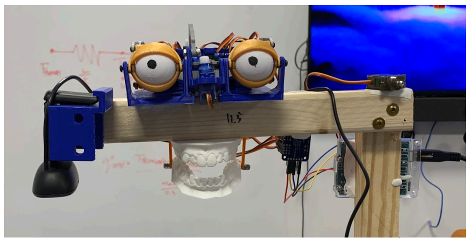

Figure 1: Animatronic Eyes, Mouth, and Camera for CV on Overhead Mount

Voice Clone

Kent Yamamoto provided the voice for our robot. We recorded samples of Kent speaking complex sentences containing a large variety of English phonemes in various positions within words and sentences. This work captured required parts of speech to effectively mimic natural language. We also recorded Kent speaking fluidly using intuitive speech, so that our cloned voice would have a realistic and natural feel.

With the cloned voice we produced over 50 pre-recorded sayings. Additionally the voice can be accessed through API by calling a Local LLM (Llamafile) or ChatGPT, and ElevenLabs for live and novel interaction.

Voice Operation

Human players are able to direct the robot's movements using voice commands by calling out squares on the checkerboard (ex. A4 B3). This ability works well in quiet environments, but struggles elsewhere.

End Effector Design

Electrical

The end effector makes use of an 8V electromagnet controlled by a 5V wired relay. The electromagnet is located inside the end effector. The relay is controlled by the Robot's GPIO pins.

Mechanical

The custom end effector was 3D modeled in Onshape and printed to friction fit into cobot's switchable end effector slots. The end effector is designed to have some give with the electromagnet in the event of direct contact between the magnet and the checkerboard. In addition to the simple friction fit, the attachment point for the power and ground wires has a small bit of strain relief, which is also utilized to help keep the electromagnet inside the end effector shell. This helps keep the electromagnet at the proper position.

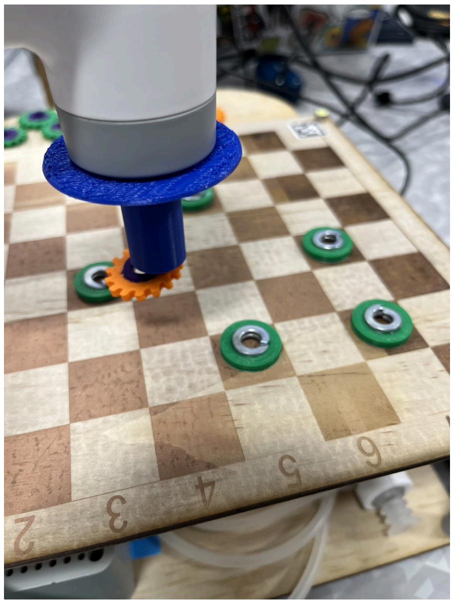

Figure 2: End Effector Moving King

Motion and Path Planning

Checker Movement Robot

There are four main actions the robot can perform: move a piece from one square to the next, move a piece off the board after a jump or kinging occurs, add a king to the board, or move out of the way of the board in order to allow a human turn without obstruction. Motion and path planning is integrated with other robotic system parts (electromagnet, voice clone, animatronics, and camera) to allow for proper operation of the entire robot.

Checker Movement in Simulation

The URDF of the mycobot robot arm was modified to include the custom designed end effector and the complete model checkerboard. The moveit2 assistant was used to generate the required RViz ROS2 URDF files. An associated python file was developed to generate ROS2 nodes with moveit2 for motion planning.

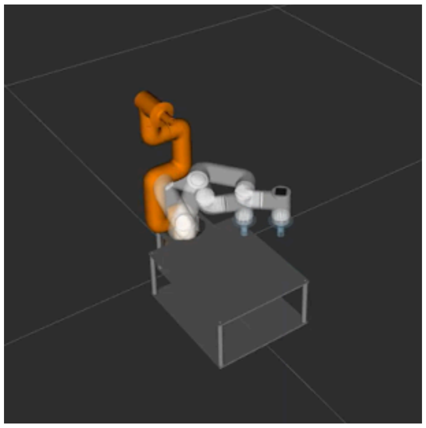

Image 3: Screenshot of RViz Robot Simulation

<u>Computer Vision</u>

The computer vision integration allows for ease in the robot recognizing human moves. The video capture allows for the game memory to be updated with the current board layout.

The computer vision module utilizes a live video feed from a camera on a stand that is detached from the board. The decision to detach the camera and stand was due to robot storage space limitations. This detachment presents an additional difficulty of allowing for a variety of camera locations and rotations in relation to the board. The difficulty creates the need for capture of not only piece locations, but also board location and orientation. This was accomplished using 4 April Tags, where only 3 are required and the last is present for redundancy.

An HSV filter with contour detection is combined with the known board orientation to identify piece color, location, and king status.

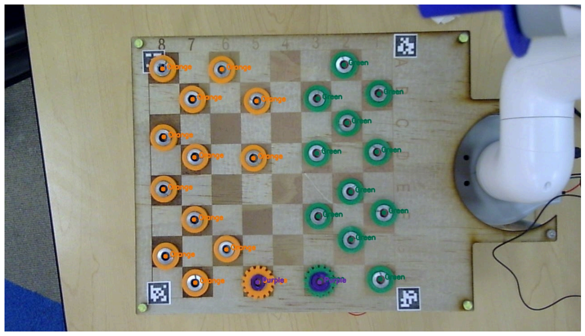

Image 4: Pieces Labelled By CV

<u>Checkers Base Code</u>

The game was built using Object Oriented Programming. Classes were created for the piece, board, and game. The piece class contained information about and methods concerning the location, color, king status, etc. The board class contained information about and methods concerning the available movements, current board layout, etc. The game class contained play for the human, random, prioritize jumps, LLM, and minimax actors.

<u>Networking and Integration</u>

The system was designed as a distributed brain system, meaning that one main brain controls other brains that branch off from it. In our system, the main brain is the laptop running the game code, LLM, and computer vision models. The other brains are the Raspberry Pi onboard the robot arm and the arduino on the animatronics. This ensures a smooth and fast system due to limited processing power on the Pi.

To integrate all these parts, we utilize TCP/IP protocol. The laptop sends strings over a socket connection to the Pi, which interprets those strings into robot commands, sending PWM signals to the servos to move them to specific angles. Meanwhile, a serial protocol is set up over usb to the laptop, integrating the animatronics into the system. The laptop sends an integer to the animatronics' arduino, which triggers the mouth servo to move for the specified duration.

Results and Discussion

Results Playing Against Human

Minimax Algorithm

This algorithm allows the robot to see several moves deep into the game, with more options than most humans would care to consider. As such, this robot regularly performs well against humans. However, an expert player will be able to beat this algorithm by considering more than 5 future moves.

Random/Prioritize Jumps

As most of the moves are random, this game mode makes many poor moves. When a jump is available, the robot opponent acts greedy. A human can exploit this behavior to set up multiple jumps for themselves. As such, a thoughtful human was able to beat this game mode consistently.

Pure Random

The pure random play is poor at best, resulting in many missed opportunities and bad decisions. This game mode easily allows for a human to win.

When a human opponent is learning to play, this method allows for the player to practice setting up jumps and traps for an opponent that isn't capable of identifying and reacting to any moves.

LLM

The LLM produces intuitive and beneficial moves nearly half the time. At the other times, the LLM produces disadvantageous moves due to poor understanding of the game and hallucinations (resulting in a failover to the prioritize jumps algorithm). Because of these poor moves, the human is able to beat the LLM on a consistent basis.

This is in line with current findings that LLMs struggle to consistently win Tic-Tac-Toe against a human opponent.

Results, Algorithm vs Algorithm

When playing itself or other algorithms, the best computer player is the minimax algorithm. The LLM's hallucinations and the randomness of the other two algorithms make them inferior in comparison.

Conclusion

This autonomous checkers playing robot is able to smoothly play against either a human or itself, as well as interact through its animatronics in a lifelike fashion.

Future work could improve upon the gameplay algorithms, incorporate additional board game options such as chess or connect 4, and upgrade the electromagnet operation to increase the success rate of planned piece movements.

Sources:

Robot Arm

https://shop.elephantrobotics.com/collections/mycobot-280

Animatronic Eyes

https://willcogley.notion.site/Will-Cogley-Project-Archive-75a4864d73ab4361ab26cabaadaec33a

3D Teeth Mold

https://makerworld.com/en/models/657234#profileId-584358

LLM Tic-Tac-Toe Difficulties

https://www.reuters.com/technology/artificial-intelligence/openai-working-new-reasoning-technology-under-code-name-strawberry-2024-07-12/