**הטכניון - מכון טכנולוגי לישראל**

**TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY**

**הפקולטה להנדסת חשמל**

**המעבדה לבקרה רובוטיקה ולמידה חישובית**

**הנושא:**

**RoboChess: Chess Playing Robotic Arm**

**Using Image Processing & Computer Vision**

**מגישים:**

מוחמד גנאים אחמד גנאים

**מנחה:**

אלי שפר

**אביב תשפ"ד**

**Table of Contents**

Abstract ……………………………………………………………………………. 3

Motivation and Introduction ………………………………………………………. 4

Physical hardware and requirements ………………………………………………. 5

Game Flow Chart ………………………………………………………………….. 6

Project Structure ………………………………………………………………… 7-8

Major Classes and Libraries ……………………………………………………… 9-11

Problems & Solutions ………………………………………………………… 12-13

Conclusions ……………………………………………………………………… 14

Setup Guide ……………………………………………………………………… 15

Future Improvements …………………………………………………………… 16

Bibliography ………………………………………………………………………. 17

**Abstract**

Chess has undergone a transformation in the digital age, becoming more accessible and popular than ever before. Online platforms like Chess.com allow players worldwide to connect and compete, fostering a global community that spans skill levels. This digital evolution has expanded chess's reach, bringing the game into the lives of millions, enhancing its appeal through online matches, tutorials, and virtual tournaments.

Our project introduces the xArm Lite6, a robotic arm engineered to play 1v1 chess, merging the physical with the digital. Designed to challenge human opponents across four difficulty levels, the xArm Lite6 enhances players' chess skills through interactive play. It offers a unique blend of mechanical precision and strategic gameplay, serving not only as an opponent but also as a tool for learning and improvement. This innovative approach aims to make chess more engaging, providing a tangible way to practice and enjoy the game in a new dimension.

**Motivation and Introduction**

Technological advancements have greatly expanded chess's reach, with platforms like Chess.com connecting players worldwide. Yet, the digital shift overlooks the tactile joy of traditional chess, creating a niche for innovation that marries the physical with the digital. Advances in image processing and object detection now enable such innovations, allowing for real-time interaction with physical objects in a game setting.

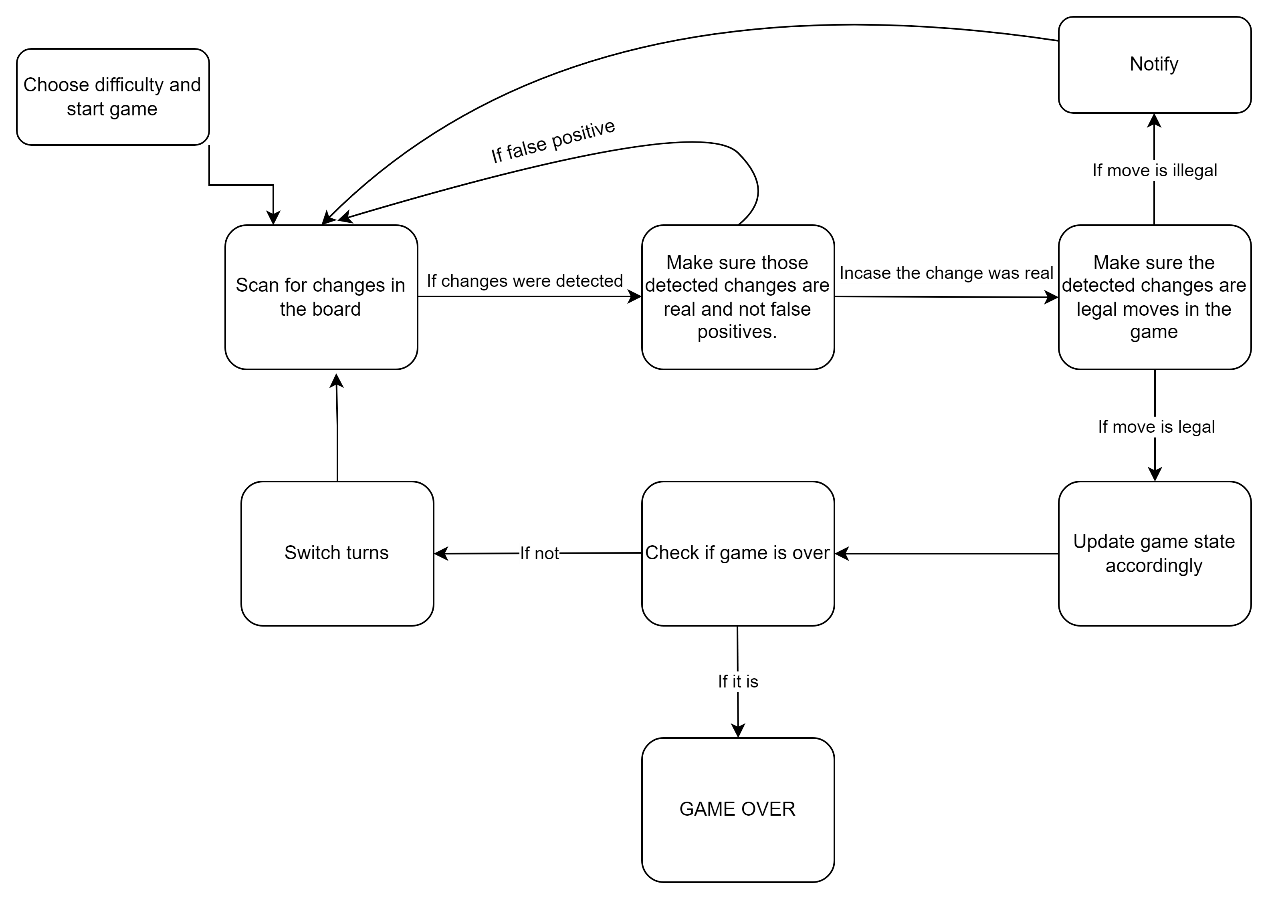
We leveraged these technologies in our project, employing the xArm Lite6 robotic arm to introduce a chess experience across four difficulty levels: Infant, Beginner, Amateur, and Grand Master. This setup is designed to cater to a broad audience, from novices to seasoned players. Central to our innovation is the application of image processing to monitor the chessboard and adapt the robotic arm's strategy dynamically. Our design includes a custom set of 3D printed cylindrical chess pieces, optimized for the arm’s vacuum gripper, ensuring smooth gameplay.

Furthermore, we developed a user interface that displays both a real-time top view of the physical chessboard and a digital board. The digital board represents the game in a familiar online format, blending the best of both physical and digital chess worlds. This approach not only fills the gap between these two experiences but also enhances player engagement and learning, offering a unique and comprehensive chess-playing solution.

**Physical hardware and requirements**

* **Robotic Arm**: The project utilizes the xArm Lite6 robotic arm, selected for its precision and flexibility. This arm is crucial for the physical aspect of our chess game, responsible for moving the chess pieces from one square to another, mimicking human-like movements to engage players in the physical space.
* **Intel's RealSense D435 Camera**: This advanced camera is key for capturing detailed visual input, utilizing depth sensing to accurately determine the positions of chess pieces on the board. Its precision is vital for the system’s ability to interact effectively with the chessboard and pieces.
* **Standard 8x8 Chessboard**: Essential for gameplay, the standard layout provides a familiar interface for both the system and the players, ensuring a traditional chess experience with a technological twist.
* **Custom-Made 3D Printed Cylindrical Chess Pieces**: These pieces are designed to be easily manipulated by the robotic arm's vacuum gripper. Their unique shape enhances the system's interaction with the pieces, ensuring stable and accurate movements.
* **Green & Yellow Tape**: Applied to the edges of the chessboard, this tape significantly improves the system's detection of the board's limits. This enhancement is crucial for accurately transforming the captured visual frame into a top-view digital representation of the board, aligning the physical game with its digital analysis.

**Game Flow Chart**



**A screenshot of a computer

Description automatically generatedProject Structure**

Our project is divided into distinct modules, making it easier to manage and understand by breaking down the overall objective into smaller, focused tasks. This modular structure reduces complexities and potential issues, ensuring a smoother development process. It allows for a more efficient approach to tackling each aspect of the project, leading to streamlined execution and effective results.

In our project's **chess** folder, we've organized the robotic arm chess game into four primary files to ensure a smooth development process and engaging gameplay. **chess\_logic.py** is the core, handling all game mechanics and rules, ensuring that the robotic arm follows the official chess guidelines. **chessboard.py** models the physical board, which is the arena for the robotic arm and human player. The **chess\_module.py** acts as a bridge, linking the game's logic with the physical interactions on the chessboard, coordinating movements between the robotic arm and the game state. For the **chess\_viz.py**, it provides a 2D visual representation of the chessboard, reflecting the game's progress and moves in real-time, making it easier for users to follow along with the robotic arm's actions. Additionally, the **PIECES** folder contains images crucial for this visualization, ensuring players have a clear and accurate view of the game as it unfolds. Together, these components combine technology and strategy to create a dynamic and interactive chess-playing experience with a robotic arm.

The **arm\_module.py** file serves as the interface for precise control over the robotic arm, crucial for the chess game's physical interactions. **camera.py** establishes communication with the camera, key for real-time game monitoring and adjustments. **GUI.py** is tasked with creating a user-friendly game interface, enriching the player's experience. In **image\_processing.py**, essential functions for processing and handling

visual data are stored, ensuring accurate game visualization. **motion\_detector.py** utilizes depth sensing to detect motion, adding responsiveness to the game environment. utilities.py compiles general helper functions that streamline project operations. Additionally, .npy and .pkl files within this folder are used for temporary storage of masks and data, supporting the game's functionality and performance.

**main.py** acts as the central hub for our project, orchestrating the entire operation by linking all the modules together. It's designed to kickstart the project, ensuring that each component—from the robotic arm's control and camera interaction to the game's user interface and image processing—works in harmony. Beyond just connecting these modules, `main.py` is responsible for creating separate threads for each task, allowing for parallel processing. This approach enhances the system's efficiency and responsiveness, enabling a smooth and integrated experience for playing chess against the robotic arm. Through main.py, we ensure a seamless flow between the physical movements of the chess pieces by the robotic arm and their digital representation on the screen, making it the cornerstone of our project's functionality.

**Major Classes**

**Chessboard**

Manages the physical representation of the chessboard within the application, including the squares, the positions of pieces on the board, and visual cues (like circles to represent possible moves or captures).

- Maintains the current and previous state of pieces on the board to track changes and moves.

- Utilizes an image processor to detect the board and initialize board parameters, such as warping for perspective correction.

- Loads and updates square coordinates for accurate piece placement and movement tracking.

**ChessLogic**

Serves as the brain of the chess game, handling game rules, move validation, and interfacing with an external chess engine for move generation and analysis.

- Initializes with a user interface manager and optionally a path to a chess engine executable.

- Manages a chess engine session for computing moves and analyzing the game state.

- Keeps track of the board's logical state and manages the timing for the engine's move thinking.

- Responsible for generating the next move to the arm.

**ChessGame**

Acts as the central coordinator for the chess game, linking the physical board representation with the game's logic and user interactions.

- Initializes with a frame, an image processor, and a user interface manager, creating instances of `Chessboard` for the physical board and `ChessLogic` for game logic.

- Manages the turn sequence between the player and the robotic arm, determining when each side makes a move.

- Integrates the user interface for real-time game updates and interactions.

- Translates the physical board state into a logical board state for the chess engine to analyze and decide on the next best move.

**ImageProcessor**

Contains Image Processing algorithms and techniques for various uses such as:

* Applying image processing filters in order to enhance our frame and get rid of noise.
* Defining masks in order to detect the chess-board in the frame.
* Dividing the chessboard into 8x8 squares and defining each square's borders.
* Detecting changes in the chessboard for proper game state update.
* Responsible for returning a bird-eye view frame of the chessboard using the original frame.

**RoboticArm**

Interfaces with the robotic arm hardware, issuing commands for piece movement. It translates the game logic into physical actions on the chessboard.

* Responsible for path planning and execution of commands for the arm.

**MotionDetector**

* Responsible for detecting movements in the frame using the depth capabilities of the camera. When such movements are detected, the scanning process of the frame is ignored.

**Problems & Solutions**

In the development of our interactive chess game, we encountered a variety of challenges that needed to be overcome to ensure a smooth and engaging experience. Below, we detail some of the primary obstacles and our innovative solutions:

* **Chessboard Detection**: The chessboard, being our region of interest (ROI), required precise detection within the frame. To achieve this, we wrapped the board's edges with yellow and green tape, creating a distinct boundary. This approach allowed us to generate a mask for detecting these colors and isolate the region within as the chessboard.yellow & green and then returned the region inside that connected component.
* **Noise Filtering**: Varied lighting conditions and camera quality often introduced noise into the frames. This was particularly troublesome during the detection of the yellow and green tape. We employed techniques such as dilation, blurring, and color space conversion to mitigate these issues and enhance image quality.
* **Piece Detection**: Instead of relying on AI for piece detection, we utilized the unique physical attributes of our 3D printed chess pieces. By applying image processing methods, including the Hough Circle algorithm, we scanned each square on the board to determine the presence of a piece, thereby avoiding the complexities of AI-based detection.
* **Bird-Eye View Conversion**: Due to constraints on camera placement, achieving a direct overhead view of the board – crucial for accurately defining square borders and preventing piece occlusion – was not feasible. We addressed this by transforming the captured frame into a bird-eye perspective, allowing us to continue our image processing efforts effectively.
* **Piece Movement**: A core aspect of our project involved moving the chess pieces based on gameplay. The cylindrical shape of the pieces posed a challenge for the standard arm gripper. We switched to a vacuum gripper, refining the pickup process by sanding the tops of the pieces to ensure a smooth operation.
* **Game State Tracking**: Without the use of object detection techniques, maintaining an accurate game state was a critical goal. We developed a code-based tracking system that monitors changes on the board, assessing whether they align with legal moves and updating the game state accordingly. This system allows for immediate detection and reaction to player moves, ensuring game integrity.
* **Arm Path Planning**: Ensuring the robotic arm navigated a safe path during play was essential to avoid collisions with objects or unintended interaction with the pieces, which could disrupt game logic. We meticulously defined paths for the arm's actions, prioritizing safety and accuracy to maintain the integrity of the game.

**Conclusions**

In our project, we successfully developed an interactive chess game where players face off against a robotic arm. This achievement isn't just about creating a new way to play chess; it's about merging the tactile, traditional experience of moving chess pieces on a board with the enhancements that technology can offer. This hybrid approach doesn't only preserve the essence of chess but also enriches the player's experience and skill development by engaging them in both the physical and digital realms of the game.

When we set out on this project, our goal was to craft a chess experience that leveraged advanced technology like object detection and AI to recognize the state of the game in real-time. However, we encountered significant hurdles, particularly in amassing a large enough dataset for object detection and the limitations of our hardware to process such data. These challenges led us to pivot from our initial strategy. Instead of relying on object detection, we turned to image processing and computer vision techniques. This adjustment allowed us to achieve our objective of real-time game state recognition without the need for complex AI, demonstrating that sometimes simplicity and creativity can lead to effective solutions.

This venture into combining robotics with the strategic depth of chess opens new avenues in game theory and technology. As robotics, AI, and machine learning continue to evolve, they offer fresh perspectives and tools for enhancing traditional games and exploring new forms of interaction between humans and machines. This project not only reflects the current state of technological advancement but also hints at the potential for future innovations in game design and interaction, encouraging further exploration in the dynamic interplay of technology and traditional game play.

**Setup Guide**

1. Setup the camera in your preferred setup and angle.
2. Place the chess board in the camera's frame in a way that shows the entire chessboard properly.
3. Press Start and choose your difficulty level; Infant, Beginner, Amateur and Grand Master.
4. Place the chess pieces in the defined initial position in the code.
5. Press Ready and start playing.

**Future Improvements**

* **Automatic Board Detection Using Object Detection**: Implement object detection to automatically identify the chessboard within the environment,
* **Automatic Board State Detection Using Object Detection**: Employ object detection techniques to accurately track the positions and types of pieces on the board, ensuring the system can understand and respond to game changes in real-time.
* **Calibration Between Robotic Arm and Camera to Avoid Fixed-Place Objects/Parameters**: Develop a dynamic calibration system that synchronizes the robotic arm's movements with the camera's perspective, enabling precise interactions with the chess pieces without relying on predefined positions such as the chessboard's location.
* **Adding Game Review After the Game is Finished to Improve Gameplay**: Introduce a feature that analyzes and presents a review of the completed game, offering players insights into their performance and strategic advice for improvement.
* **Automatic Pieces Setup Using the Arm**: Upgrade the robotic arm to autonomously set up the chessboard at the start of each game, further automating the process and enhancing the user experience by preparing the board for play without human intervention.

**Bibliography**

* <https://github.com/Elucidation/ChessboardDetect/blob/master/README.md>
* <https://github.com/rjgoodloe/ESE205-CVChess>