IMPLEMENTATION OF A NAO HUMANOID ROBOT FOR TIC-TAC-TOE GAMEPLAY

BY

Mahdi Osman

Final Year project submitted in partial fulfilment of the requirements for the Degree of

Bachelor of Science in Computer Science

Department of Computer Science

School of Sciences and Engineering

University of Nicosia

June 2024

Implementation of a NAO Humanoid Robot for Tic-Tac-Toe Gameplay

By

Mahdi Osman

This Final Year Project has been accepted in partial fulfilment of the requirements for the Degree of

Bachelor of Science in Computer Science

Project Advisor:

_____ _____ ___ / ___/ _

(Name) (Signature) (Date)

Examiner:

_____ _____ ___ / ___/ _

(Name) (Signature) (Date)

Examiner:

_____ _____ ___ / ___/ _

(Name) (Signature) (Date)

Department of Computer Science

School of Sciences and Engineering

University of Nicosia

Abstract

This project explores the development and implementation of a NAO humanoid robot designed to play Tic-Tac-Toe against a human opponent. The primary objective is to integrate computer vision and artificial intelligence to enhance the robot's interactive capabilities. By leveraging the OpenCV library, the project enables the NAO robot to perceive and interpret its environment, focusing on computer vision, and making informed decisions based on visual stimuli.

The methodology includes implementing computer vision algorithms to process visual data captured by the robot's cameras, allowing it to recognize the Tic-Tac-Toe board and the moves made by the human player. The project also implements the minimax algorithm with alpha-beta pruning to enable the robot to make optimal moves during gameplay. The NAOqi SDK and Python programming are utilized to ensure seamless integration with the robot's hardware and software, facilitating effective human-robot interaction.

Extensive testing and evaluation demonstrate the system's ability to accurately identify and respond to game states, showcasing the NAO robot's potential for engaging and interactive gameplay. The project addresses challenges related to real-time processing and dynamic environment adaptability, offering insights into the practical application of computer vision and AI in humanoid robotics.

Acknowledgements

I would like to extend my heartfelt gratitude to several individuals who have supported and guided me throughout the completion of this thesis project.

First and foremost, I would like to thank my parents for their unwavering encouragement and financial support during my studies. Their belief in my abilities has been a constant source of motivation, and I am truly grateful for their sacrifices.

I am indebted to my project advisor, Dr. Athena Stassopoulou, for her invaluable guidance, insightful feedback, and critical questions that challenged me at every stage. Her mentorship has not only enhanced the quality of this project but also enriched my learning experience, teaching me the importance of critical thinking and clarity in writing.

I would like to add a special mention to Mr. Ali Tayari, whose unwavering support and expertise were invaluable throughout the development process of this project. Mr. Tayari's willingness to assist with every problem I encountered, whether technical or conceptual, significantly contributed to the success of this project. His guidance, patience, and dedication were instrumental in overcoming challenges and achieving milestones.

Lastly, I would like to thank all the participants, colleagues, and individuals who contributed to this project in various ways, whether through collaboration, feedback, or moral support. Your contributions have been invaluable in shaping the outcome of this research.

Table of Contents

List of Illustrations

Page

Chapter 1 Introduction

1.1 Background

In recent years, the rapid progression of robotics has paved the way for transformative shifts. All types of robots, ranging from industrial arms in manufacturing to social robots in elder care, are progressively becoming integral components across various sectors, including healthcare, education, hospitality, and beyond.

In the field of healthcare, advancements in robotic radiation technology, such as the CyberKnife System [2], which offers non-invasive treatment for cancerous and non-cancerous tumours. Innovative systems such as this empower surgeons to execute intricate surgeries with heightened accuracy and dexterity, guiding a new era of surgical precision and patient care.

Additionally, social robots like NAO and Pepper have found applications in education, where they serve as interactive teaching assistants, engaging students in immersive learning experiences [19]. Through natural language processing and facial recognition capabilities, Pepper can tailor its interactions to individual students' needs, providing personalized support and feedback.

Moreover, the NAO robot, renowned for its humanoid design and versatility, stands as a prominent example in this evolving landscape of robotic technology. With its expressive movements and interactive features, NAO has been employed in various domains, from research and therapy to customer service and entertainment, offering unprecedented opportunities for collaboration between humans and machines [1].

These advancements underscore the potential of robots to augment human capabilities, enhance productivity, and improve quality of life across diverse contexts, marking a significant paradigm shift in how we interact and collaborate with technology.

1.2 Motivation

As we navigate the intricacies of enhancing HRI, the integration of computer vision emerges as a pivotal direction for augmenting a robot's perceptual capabilities. Computer vision, a multidisciplinary field at the intersection of computer science and image processing, equips robots with the ability to interpret and respond to visual stimuli from their surroundings. This embarks on the exploration of leveraging computer vision techniques to empower the NAO robot, contributing to the broader objective of create more intuitive human-robot interactions.

1.3 NAO

1.3.1 Background

Project NAO began in 2004 with the ambitious goal of creating an autonomous, programmable humanoid robot. The initiative was spearheaded by Aldebaran Robotics[1] (formerly known as SoftBank Robotics), a pioneering French company based in Paris. This project marked the beginning of what would become a significant advancement in humanoid robotics [30].

1.3.2 Features

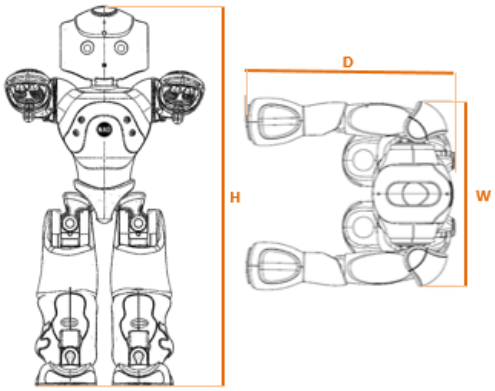
The NAO robot is filled with features that make it a versatile and engaging companion. Standing at 58cm tall, NAO has a humanoid design with 25 degrees of freedom,

[1] https://www.aldebaran.com/

enabling fluid and lifelike movements. Sophisticated sensors, including tactile sensors, sonar, cameras, and inertial movement sensors, allow it to perceive its environment and interact with users effectively. Its speech recognition and synthesis capabilities facilitate natural language interaction, while its high-definition cameras and microphone array enable it to see, hear, and respond to its surroundings. Additionally, NAO's Wi-Fi and Ethernet connectivity enable seamless integration into various networks and systems, making it an ideal platform for research, education, and entertainment. With its customizable programming framework and extensive software ecosystem, NAO offers endless possibilities for developers and enthusiasts to create innovative applications and experiences. All of NAO's features can be seen in Fig. 1.1-1.6

Figure 1.1: NAO Robot[2]

[2] http://doc.aldebaran.com/



Height (mm)	Depth (mm)	Width (mm)
574	311	275

Figure 1.2: NAO Dimensions[3]

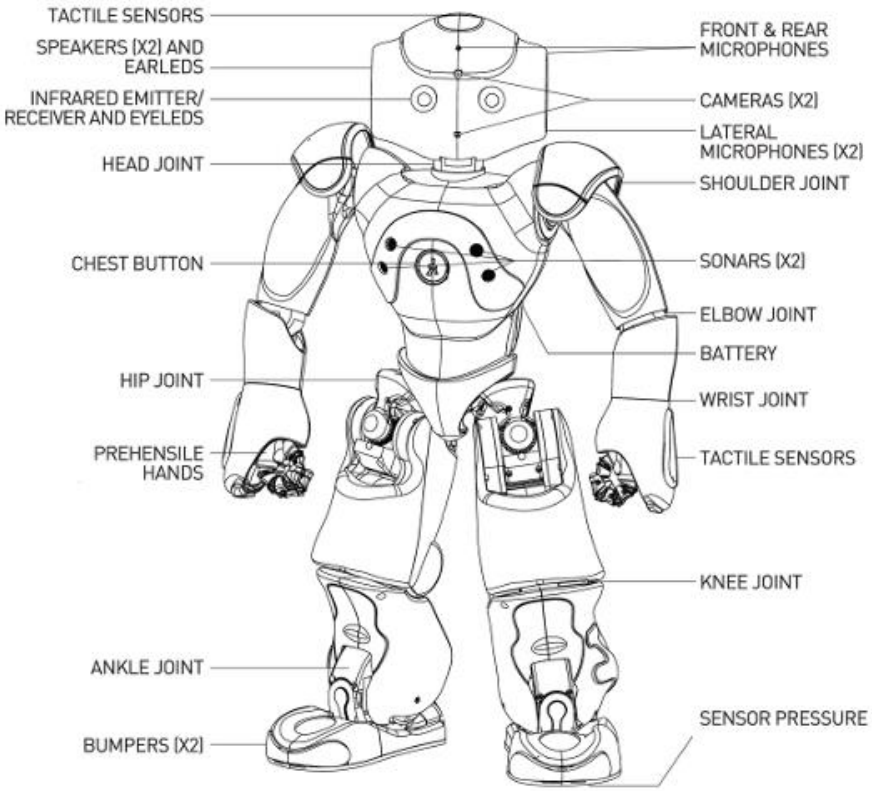


Figure 1.3: NAO Anatomy[3]

[3] http://doc.aldebaran.com/

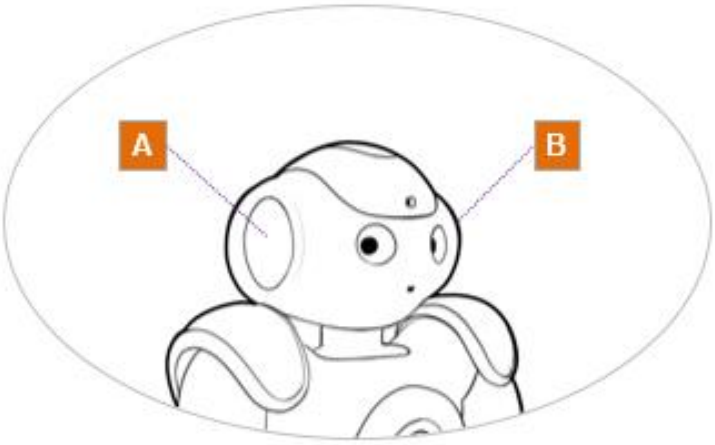


Figure 1.4: NAO Speakers[4]

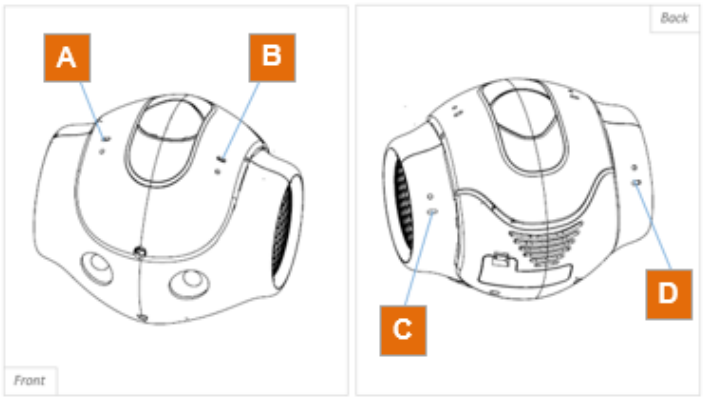


Figure 1.5: NAO Microphones[4]

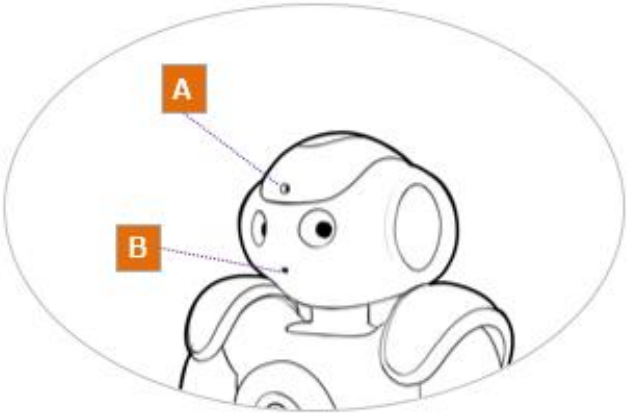


Figure 1.6: NAO Cameras[4]

[4] http://doc.aldebaran.com/

1.3.3 Choreographe

Choreographe[5] is a comprehensive software suite designed to empower users to program and control the NAO robot with ease. Choreographe allows users to create complex behaviours and interactions through drag-and-drop programming, eliminating the need for extensive coding knowledge. Users can design dynamic movements, gestures, and expressions for NAO using the timeline-based editor, enabling precise control over its actions. Choreographe also provides access to NAO's sensor data, enabling users to create reactive behaviours based on real-time input. Additionally, Choreographe supports the integration of speech recognition and synthesis, enabling NAO to understand and generate natural language. Furthermore, Choreographe offers advanced features such as simulation mode, allowing users to test and debug their programs before deploying them to the physical robot. With its user-friendly interface and powerful capabilities, Choreographe is an indispensable tool for unlocking the full potential of the NAO robot in various applications, including education, research, and entertainment.

1.3.4 NAOqi SDK

The NAOqi SDK[6] provides developers with a comprehensive set of tools and APIs to create advanced applications for the NAO robot, supporting multiple languages - including Python, C++, and JAVA. The SDK offers access to NAO's sensors, actuators, and behaviours, that enable the creation of custom behaviours and interactions tailored

---

[5] https://www.aldebaran.com/en/support/nao-6/downloads-softwares

[6] http://doc.aldebaran.com/2-8/dev/programming_index.html

to specific use cases. Through the SDK, we can utilize NAO's speech recognition and synthesis capabilities, enabling natural language interaction with the robot. Additionally, the SDK provides tools for accessing and processing sensor data, enabling us to create reactive behaviours based on the robot's perception of its environment. Furthermore, the NAOqi SDK includes simulation tools, allowing us to test and debug their applications in a virtual environment before deploying them to the physical robot. The NAOqi SDK offers a comprehensive set of tools and resources, supported by thorough documentation and a vibrant developer community. This makes it an excellent platform for developing creative and innovative applications for the NAO robot, in various domains such as education, healthcare, and entertainment, among others.

1.4 Objectives

The primary objective of this project is to develop and implement a robust system that integrates computer vision capabilities into the functionality of the NAO robot to play a Tic-Tac-Toe game against a human opponent. Through the utilization of OpenCV[7], a widely adopted computer vision library, the aim is to enable the NAO robot to perceive, and understand, its visual environment. Additionally, the project makes use of the minimax algorithm to ensure the NAO robot can respond intelligently to the game's state. The project seeks to address the following key objectives:

- To implement computer vision algorithms on the NAO robot for object recognition, tracking, and visual interpretation.

[7] https://opencv.org/

- To design a human-robot interaction where the human makes a physical move on a tic-tac-toe board and the robot responds through verbal announcements and screen displays.
- To implement a game-playing component for the NAO robot to enable it to make game decisions and play optimally against a human opponent.
- To evaluate the resulting system by assessing the user experience by testing it on several human opponents and assessing NAO's accuracy in detecting the moves, in choosing its own optimal, the robot's feedback (verbal and visual) and the user engagement and satisfaction.

In addition, as a final objective of the project is to contribute insights into the challenges and opportunities associated with integrating computer vision into humanoid robotic platforms.

1.5 Thesis Structure

The rest of the thesis is structured as follows: In Chapter 2, a review of the published literature on NAO and its applications is presented. In Chapter 3, the overall system architecture and design are described. Chapter 4 describes the communication protocols and methods used in the system. Chapter 5 focuses on the computer vision techniques implemented. Chapter 6 discusses the development of the game-playing modules of the NAO robot. Chapter 7 presents the results of the experiments and provides an analysis of the findings. Finally, Chapter 8 summarizes the work and discusses potential future directions for research.

Chapter 2 Literature Review

2.1 Introduction

The integration of computer vision and artificial intelligence (AI) into humanoid robots has introduced a new era of robotics research. These advancements hold immense promise across diverse domains, including healthcare, education, and entertainment. In this literature review, we delve into existing research, cutting-edge technologies, and methodological approaches related to this integration, with a specific focus on the NAO robot. By critically assessing the current state of the field, identifying knowledge gaps, and establishing a robust theoretical framework, we pave the way for deeper insights and practical applications.

2.2 Computer Vision Technologies

Computer vision technologies serve as the eyes of humanoid robots, enabling them to perceive and interpret visual information from their surroundings. Within this context, the NAO robot benefits from various algorithms and techniques.

2.2.1 Object Recognition

Object recognition algorithms empower the NAO to identify and categorize objects in its environment. By analysing visual cues, the robot can distinguish between everyday items, recognize faces, and interact with its surroundings intelligently. Noteworthy projects in this area include real-time object detection and classification using deep neural networks [14]. According to this work, the robot recognized about 90 everyday objects, with an average processing time of about 100 milliseconds. "*Real-Time Object Recognition Based on NAO Humanoid Robot*" focuses on the real-time uses of object recognition for indoor humanoid robots such as NAO [37]; this paper mainly focuses

on making the NAO robot detect objects faster and more efficiently, so that the entire process can be streamlined.

2.2.2 Image Processing

Image processing techniques enhance the NAO's ability to process visual data efficiently. From noise reduction to feature extraction, these methods contribute to accurate perception. Researchers have explored image pyramid segmentation and deep learning approaches to optimize image analysis [13]. The work done here shows the precise position of any target object can be deduced in the world frame. By implementing advanced image processing methods, the NAO robot can better interpret its surroundings, leading to more effective and reliable interactions with humans.

2.2.3 Visual Interpretation

Visual interpretation involves extracting meaningful information from images. For the NAO, this translates into understanding gestures, expressions, and contextual cues. By leveraging computer vision, the robot can engage in natural interactions with humans, making it a valuable tool in educational settings, healthcare, and beyond [3]. In this work, an in-depth scoping review of around 300 studies on the use of the NAO robot in human-robot interaction from 2010 to 2020, providing qualitative and quantitative insights into its applications, capabilities, communication methods, and experimental designs.

In summary, computer vision technologies are pivotal for the NAO robot's success. As we delve deeper into this field, we must address challenges related to biological motion mechanisms, structural design, material applications, and energy efficiency. The integration of bionics, brain-inspired intelligence, mechanics, and control holds immense promise for advancing humanoid robotics.

2.2.4 Challenges and Opportunities

Integrating computer vision into humanoid robots like NAO presents challenges such as real-time processing constraints and adaptability in dynamic environments. Research highlights the importance of addressing locomotion, awareness, decision-making, and robustness to advance the field [21]. This paper introduces an innovative and engaging methodology for teaching vision systems using the NAO humanoid robot. The integration of bionics, brain-inspired intelligence, mechanics, and control is identified as a promising direction for overcoming these challenges [28]. This work shows the development and real-time implementation of a bioinspired adaptive model using a spiking neural network to control a humanoid NAO robot, successfully replicating human-like adaptive motor control in response to external perturbations. Addressing these challenges requires the development of more efficient algorithms and hardware capable of handling complex visual data in real-time. Additionally, there is a need for further research into making these systems more adaptable to changing environments and user needs.

2.3 Human-Robot Interaction Frameworks

The field of human-robot interaction (HRI) has seen the development of numerous frameworks aimed at enhancing the interplay between humans and robots. A notable contribution is the work of [Rabb et. Al][23] “Attachment Framework for Human-Robot Interaction”, which explores the psychological aspects of attachment in HRI, providing a spectrum to gauge attachment potential and actual engendered levels of attachment in study settings. Understanding these psychological aspects is crucial for developing robots that can form meaningful connections with humans, which is essential for applications in therapy, education, and personal assistance.

Additionally, the work of [Tanevska et. al][27] "*Socially Adaptable Framework for Human-Robot Interaction*" emphasizes the adaptability of robots to human affective states, enhancing the richness of HRI. This framework is important for creating robots that can adjust their behaviour based on the emotional state of the user, making interactions more natural and effective. The study in [24] examines the features and capabilities of the NAO robot, including its motricity, functionality, and affective capacities, across different contexts. It also reviews research that has utilized the NAO robot to explore its potential as a Socially Assistive Robot. These studies provide valuable insights into the design and implementation of robots that can assist in social and therapeutic settings, highlighting the versatility and potential of the NAO robot in various applications.

2.3.1 NAO as a Teaching Assistant

Recent research explored the impact of the NAO robot as a teaching assistant on university students' vocabulary learning and their attitudes towards this technological tool [29]. The study employed a mixed-method approach to collect both quantitative and qualitative data. A quasi-experimental design, incorporating pre-tests and post-tests, was used to assess the effectiveness of the NAO robot on students' vocabulary acquisition. Additionally, questionnaires and interviews provided insights into participants' attitudes towards the robot. The participants were freshman students in an English language teaching undergraduate program at a higher education institution, engaged in a vocabulary course as part of their curriculum. Findings indicated that, although the control group slightly outperformed the experimental group, there was no statistically significant difference between them. Descriptive analysis suggested that most students had a favourable view of the NAO robot and appreciated its capabilities. Nonetheless, qualitative data revealed mixed opinions, with many participants

acknowledging the robot's role in aiding vocabulary learning, while some noted the need for technological enhancements. This study highlights the potential of robotic teaching assistants in language education, although further refinement and research are necessary to optimize their effectiveness [37].

2.4 Summary

The NAO robot, developed by Aldebaran Robotics, is a versatile platform used in education, research, and healthcare. It can interact with people on a personal level, speaking multiple languages, and navigating its environment [1]. Its applications range from serving as a teaching assistant to participating in interactive experiments and data collection for research purposes [10]. The integration of computer vision technologies into the NAO robot enhances its capabilities, enabling it to perceive and interpret its environment more effectively.

This chapter presented published literature on the integration of computer vision into humanoid robots, particularly focusing on the NAO robot. It identifies gaps in knowledge and outlines the potential for advancements in HRI frameworks, the capabilities and applications of the NAO robot, and the challenges and opportunities in the field. The insights gained lay the groundwork for future research, aiming to enhance the functionality and applicability of humanoid robots. By addressing the identified challenges and leveraging the opportunities presented by advanced technologies, researchers can continue to push the boundaries of what humanoid robots can achieve, making them more useful and accessible in a variety of settings.

Chapter 3 System Overview

3.1 Overview

In this chapter, we give an overview of the system and its various modules. The main modules of the project are:

1) The Vision module, which is responsible for capturing and processing images of the game board.

2) The Game playing module, which determines the optimal moves and updates the game state.

3) The Move execution module, that communicates moves orally and displays them.

An overview of each of these modules is presented in the following subsections. The flowchart of the system is shown in Figure 3.1. The process starts with camera calibration to ensure all systems are properly set up. Following this, the first player is chosen, which can be either the human or the NAO robot. If the human is selected to go first, they make a move on the physical game board. The system then captures an image of the board, processes it to recognize the human's move, and updates the game state accordingly. The system checks if the human has won the game. If the human wins, NAO announces the human as the winner. If not, the game algorithm determines the optimal move for NAO, which is then announced and displayed on the computer. The system checks if NAO has won. If NAO wins, it announces itself as the winner; if not, the process loops back to capturing the game board image for the next human move. Conversely, if NAO is chosen to play first, the game algorithm determines its optimal move, which is then announced and displayed. The system checks if NAO has won and, if so, announces NAO as the winner. If NAO does not win, the human makes their move, and the system captures and processes this move, updates the game state, and

checks for a human win. This iterative process of move-making, board state updating, and win-checking continues in a loop until either the human or NAO wins the game, upon which NAO announces the winner.

Figure 3.1: Process Flowchart

In the remaining sections of this chapter, we give an overview of the major processes implemented in this project. The implementation details of the processes are covered in depth in Chapter 4.

3.2 Calibration

Calibration for a tic-tac-toe board using a camera involves positioning the camera to capture a clear and optimal view of the board. The process ensures that the camera correctly identifies the grid and refines its parameters based on the environmental variables such as lighting, glare, etc. This typically includes detecting the edges of the board, identifying the lines that form the grid, and mapping the grid coordinates to image coordinates. This calibration step ensures accurate recognition of the 9 board segments during the game.

3.3 Image Processing and Move Recognition

This process involves using NAO's camera to capture images of the board, enabling the detection of moves and the updating of the game state. A brief overview of the various modules involved in this process is provided in this section.

3.3.1 Image Pre-processing

After the board is detected, the image pre-processing module prepares the image for subsequent analysis. It isolates the region of interest containing the game board, converts the image to grayscale, and employs advanced techniques to enhance edge information, enabling accurate board state detection.

Region of Interest (ROI) Extraction

This process determines the region of the image containing the game board and isolates it from the background or other irrelevant elements. This ensures that only the relevant portion of the image is processed.

Grayscale Conversion

Converts the image to grayscale to simplify subsequent processing steps while retaining essential information about edges and shapes.

Edge Enhancement

Utilizes various filters and algorithms to enhance edge information within the image. Techniques such as edge detection algorithms (e.g., Sobel, Canny) are employed to highlight the boundaries of the board and grid lines.

Noise Reduction

Removes noise or artifacts from the image that may interfere with subsequent analysis. This can involve applying filters such as Gaussian blur or morphological operations to smooth the image.

3.3.2 Board Detection

The board detection module is essential for accurately locating the game board within captured images. It employs various techniques such as contour detection and geometric shape analysis to identify the board's position and separate each grid. This process ensures consistent and reliable board localization across different images.

Contour Detection

Utilizes image processing algorithms to detect contours, or outlines, of shapes within the image. The contours of the board are identified based on their shape and characteristics.

Geometric Shape Analysis

Analyses the detected contours to identify geometric shapes that resemble the game board. This analysis may involve checking for rectangular shapes with specific aspect ratios and corner angles.

Board Localization

Once the contours resembling the board are identified, the module determines the precise location of the board within the image, considering any perspective distortion or rotation.

Grid Separation

After locating the board, the module further divides it into individual grid cells. Each cell corresponds to a potential position for a game piece, facilitating subsequent analysis of the game state.

3.3.3 Move Recognition

The player move detection module interprets the pre-processed image to determine the current state of the game board, including the position of each symbol. It converts the processed states into a structured representation, such as a 2D array. Additionally, it monitors the game space for any changes indicating a player's move.

State Interpretation

Analyses the pre-processed image to interpret the positions and identities of game pieces ('X' or 'O'). This involves recognizing patterns and shapes within the grid cells.

Structured Representation

Converts the interpreted game state into a structured format, such as a 2D array or a data structure representing the board and the positions of player pieces.

Move Monitoring

Compares consecutive frames of the game board to detect any alterations, indicating a player's move. This involves tracking changes in the positions or identities of game pieces between frames.

3.4 Game Algorithm

NAO's choice of move is enabled by the implementation of a game-playing algorithm suitable for adversarial zero-sum games, which is the minimax algorithm.

3.4.1 The Minimax Algorithm

A crucial component of the decision-making process, the minimax algorithm analyses possible future moves or game states to determine the most optimal move for the robot to make. By recursively evaluating potential moves, it selects the best strategy to maximize the robot's chances of winning.

Future Move Analysis

Considers possible future moves by simulating different sequences of actions that both the robot and the opponent could take and assigns a score to each potential game state based on an evaluation function that estimated the desirability of each state. This score is used to determine the best move for NAO.

Alpha-Beta Pruning

This algorithm optimizes the search process by eliminating branches of the game tree that are guaranteed to be worse than previously examined branches, reducing the computational complexity of the search.

3.5 Robot Move Execution

Once the next best move is determined, the robot move execution module communicates the move verbally to the user. Currently, it utilizes text-to-speech to inform the user where its next move should be played. It also displays the Tic-Tac-Toe board with the current game state on the connected computer.

Communication

Informs the user of the robot's next move using a suitable communication method, in our case as text-to-speech and visual display.

Feedback

Provides feedback to the user to confirm the execution of the move and maintain interaction during the game.

Error Handling

Handles any errors or inconsistencies that may arise during move execution, ensuring smooth gameplay experience.

Chapter 4 Communication

4.1 Introduction

NAOqi[8] serves as the core of operations for the NAO robot, controlling its functionalities and behaviours. It's a comprehensive SDK (Software Development Kit) designed to control every aspect of the robot's operations, ranging from basic movements to complex interactions with humans. It was used as the primary interface between users and the robot, NAOqi facilitates seamless communication and coordination, enabling users to send commands, receive data, and manage the robot's behaviour effectively.

4.2 Communication Protocols

NAOqi offers a versatile set of communication protocols that enable users to interact with the robot in various ways. These protocols serve as channels for transmitting commands, receiving data, and controlling the robot's actions remotely. Among the common communication protocols supported by NAOqi are HTTP, WebSocket, and TCP/IP. Each protocol offers distinct advantages and can be used based on specific requirements and preferences, providing flexibility and adaptability in communication.

Since we use a router as a hub for connection to the robot, TCP/IP is essential as it is the underlying protocol that enables communication over Wi-Fi [34].

[8] http://doc.aldebaran.com/2-8/dev/programming_index.html

4.3 Establishing a Connection

In the context of NAOqi, each module is accessed through a proxy object. A proxy acts as an intermediary or a representative for a specific module or service within the NAOqi framework. It encapsulates the communication details required to interact with the module, such as its name, IP address, and port number.

The proxy works like this:

1. When interacting with a module, developers instantiate a proxy object that represents that module. This proxy object provides a high-level interface for accessing the module's functionalities without needing to handle low-level communication details directly.
2. Each proxy object is instantiated with specific connection details, such as the module's name, IP address, and port number. These details ensure that the proxy can establish a connection with the corresponding module on the robot.
3. Each proxy object is independent and isolated from other proxies, representing a distinct module or service within the NAOqi framework. This isolation ensures that interactions with one module do not interfere with or impact the functionalities of other modules. It also promotes encapsulation, as developers we can interact with modules through well-defined interfaces provided by the proxies, without needing to be concerned about the underlying implementation details.
4. Once a proxy object is instantiated, developers can access the functionalities provided by the corresponding module using methods exposed by the proxy. These methods typically correspond to the operations supported by the module, allowing developers to send commands, retrieve data, or configure settings as needed.

5. Proxy objects also handle error conditions and exceptions transparently, abstracting away the complexities of network communication and error handling from the developer. This simplifies the development process and enhances the robustness of the application by providing consistent error handling mechanisms across different modules.

An example of how the NAO robot can be initiated can be seen in Fig. 4.1.

```
def __init__(self, ip_address, port_number):
        # Initialize NAOQI proxies
        awareness_proxy = ALProxy("ALBasicAwareness",
  ip_address, port_number)

        motion_proxy = ALProxy("ALMotion", ip_address,
  port_number)

        posture_proxy = ALProxy("ALRobotPosture",
  ip_address, port_number)

        behavior_manager_proxy =
  ALProxy("ALBehaviorManager", ip_address,
  port_number)


        self.camera_proxy = ALProxy("ALVideoDevice",
  ip_address, port_number)
```

Figure 4.1: Initiating a connection with the NAO Robot

Chapter 5 Computer Vision Module Implementation

Before we can work on detecting and or analysing any data from an image it is crucial to ensure that the image is presented in such a way that only the necessary data is seen by the software. This allows for an increase in performance as well as more time efficient processing. Since NAO needs to see the board for it to react to a players' moves, this is one of the most important modules for playing Tic-Tac-Toe.

5.1 OpenCV: A Framework for Computer Vision

5.1.1 Overview and Architecture

OpenCV[9] is a popular open-source library for computer vision, providing a wide range of algorithms and tools for image processing, object detection, and machine learning. Its modular architecture allows developers to build custom vision applications by leveraging pre-built components and libraries. OpenCV supports various programming languages, including C++, Python, and Java, making it accessible to a diverse community of developers [26]. The library's extensive documentation and active community support contribute to its widespread adoption in both academic research and industrial applications [20].

5.1.2 Integration Challenges and Solutions

Integrating OpenCV with robotics platforms presents several challenges, including hardware compatibility, real-time processing constraints, and communication protocols. Robotics platforms such as the NAO robot may have limited computational

[9] https://opencv.org/

resources and specific hardware requirements for running computer vision algorithms efficiently. Real-time processing constraints necessitate optimizing algorithms for low latency and high throughput to enable responsive interactions with the environment. Communication protocols such as ROS (Robot Operating System) facilitate seamless integration between OpenCV and robotics platforms, enabling data exchange and control commands between different components of the system. Overcoming these challenges requires careful consideration of hardware capabilities, algorithm design, and system architecture to ensure robust and reliable performance in real-world applications.

5.2 Object Recognition

Object recognition is a crucial task in computer vision, enabling robots to identify and categorize objects in their environment [9]. Traditional approaches to object recognition rely on feature extraction and matching techniques, where distinctive features of objects are detected and compared against a database of known objects. Recent advancements in deep learning have revolutionized object recognition, with convolutional neural networks (CNNs) achieving state-of-the-art performance on various recognition tasks [6]. CNNs learn hierarchical representations of objects from raw pixel data, enabling robust and accurate recognition across diverse environments.

5.3 Image Processing

Image processing forms the foundation of computer vision, encompassing techniques for manipulating and analysing digital images. Basic operations such as filtering, enhancement, and segmentation are essential for preprocessing images before further analysis. Filtering techniques like convolution and morphological operations are commonly used to remove noise and highlight relevant features in images [12].

Enhancement techniques such as histogram equalization and contrast stretching improve the visual quality of images. Segmentation techniques, including thresholding and region-based segmentation, partition images into meaningful regions for object detection and recognition [22].

5.3.1 Region of Interest

The Region of Interest or ROI is a fast way of removing unwanted noise from the image by focusing on specific areas of the image where the most important features can be seen. This can be achieved by applying any number of techniques. The simplest solution is by taking the resolution of the camera and removing a fixed area around the image, this is the simplest way to implement a ROI but requires more work by the user/player setting up the NAO robot to ensure that the grid is always visible in the centre of the image, this can be seen in Fig. 5.1.

```
8       self.roi_top_left = (90, 40)
9       self.roi_bottom_right = (550, 450)
10      self.cell_width = 145
/
...
102     roi_circles =
   self.detect_circles(image[self.roi_top_left[1]:self.
   roi_bottom_right[1],
   self.roi_top_left[0]:self.roi_bottom_right[0]])

103     if roi_circles is not None:
104          for (x, y, r) in roi_circles:
105              x += self.roi_top_left[0]
106              y += self.roi_top_left[1]

107              cv2.circle(original_image, (x, y), r,
   (0, 0, 255), 4)
108              cv2.rectangle(original_image, (x - 5, y
   - 5), (x + 5, y + 5), (0, 0, 255), -1)

109     cv2.rectangle(original_image,
   self.roi_top_left, self.roi_bottom_right, (0, 255,
   0), 2)
```

Figure 5.1: Image Region of Interest (ROI)

The ROI can also be determined by calibrating the camera before you start playing and then dynamically focusing on calibrated area. This can allow for a more accurate ROI but will require a greater amount of work to implement and is required anytime NAO is moved to a new area. The implementation of this can be seen below in Fig. 5.2:

```
    # Get the bounding box of the largest contour
345    x, y, w, h = cv2.boundingRect(max_contour)

    # Define ROI coordinates
346    roi_x1 = x
347    roi_y1 = y
348    roi_x2 = x + w
349    roi_y2 = y + h

350 return (roi_x1, roi_y1, roi_x2, roi_y2)
```

Figure 5.2: Retrieving the ROI by calibration

5.3.2 Greyscale

Converting an image to greyscale involves transforming an RGB/colour image into a single-channel image where each pixel represents the intensity of the light, ranging from 0 (black) to 255 (white). This process typically involves taking the weighted sum of the red, green, and blue channels (i.e. RGB Values) of the coloured image. OpenCV will typically use the following weighted values to convert each colour pixel in the image into a single shade of grey:

$$Grey = 0.299R + 0.587G + 0.114B$$

Greyscale images are preferred due to their simplicity and reduced computational load. By eliminating colour information, we can reduce memory and processing power. This is a lot more suitable when working with real-time data as well as a low power CPU.

OpenCV provides a simple way to convert images to greyscale using the `cv2.cvtColor()` function. This takes the input image and a conversion flag as

arguments. The conversion flag `cv2.COLOR_BGR2GRAY` converts a BGR (standard colour format in OpenCV) to greyscale as seen in Fig. 5.3.

```
# Read the image
image = cv2.imread('color_image.jpg')

# Convert the image to grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Display the original and grayscale images
cv2.imshow('Original Image', image)
cv2.imshow('Grayscale Image', gray_image)
```

Figure 5.3: Transformation of RGB to Greyscale

An example colour to greyscale conversion can be seen in Fig. 5.4.

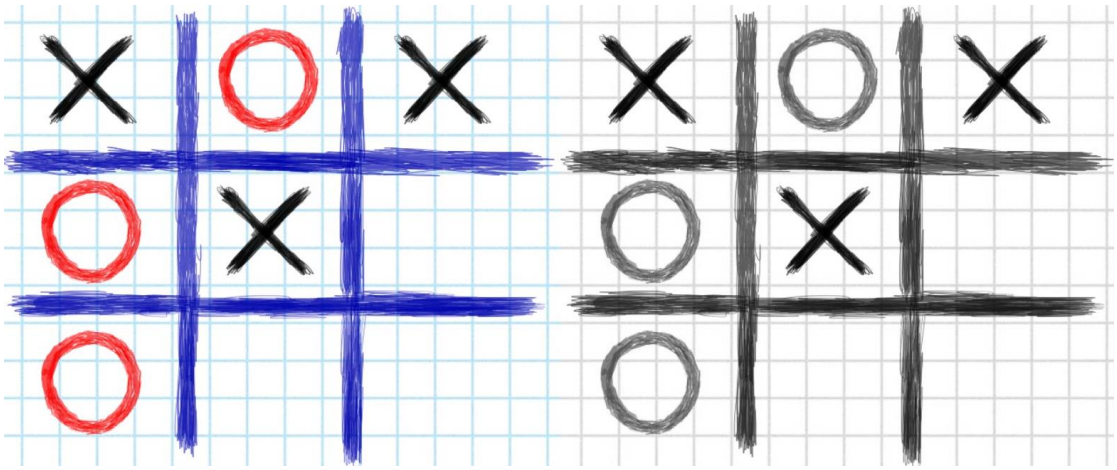


Figure 5.4: RGB to Greyscale

5.3.2 Gaussian Blur

Gaussian Blur is an essential technique in image processing used to reduce noise and detail in images. It applies a weighted average to each pixel in the image, with the weights being determined by the Gaussian distribution.

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where:

- x is the independent variable (usually representing distance from the mean).
- μ is the mean or average value.
- σ is the standard deviation, which controls the spread of the curve.

```
36          blurred = cv2.GaussianBlur(
              image,
              (15, 15),  # Kernel size (width and height)
              10.0  # Gaussian kernel standard deviation
    in X direction
          )
```

The size of the Gaussian kernel, represented by a *n x n* matrix, determines the amount of blurring applied to the image. A larger kernel size results in more extensive smoothing, while a smaller kernel size preserves finer details. Additionally, the standard deviation (σ) controls the spread of the weights. Higher values of σ lead to broader, more gradual changes in intensity and more significant blurring. The values in the code above can be seen in Fig. 5.5.

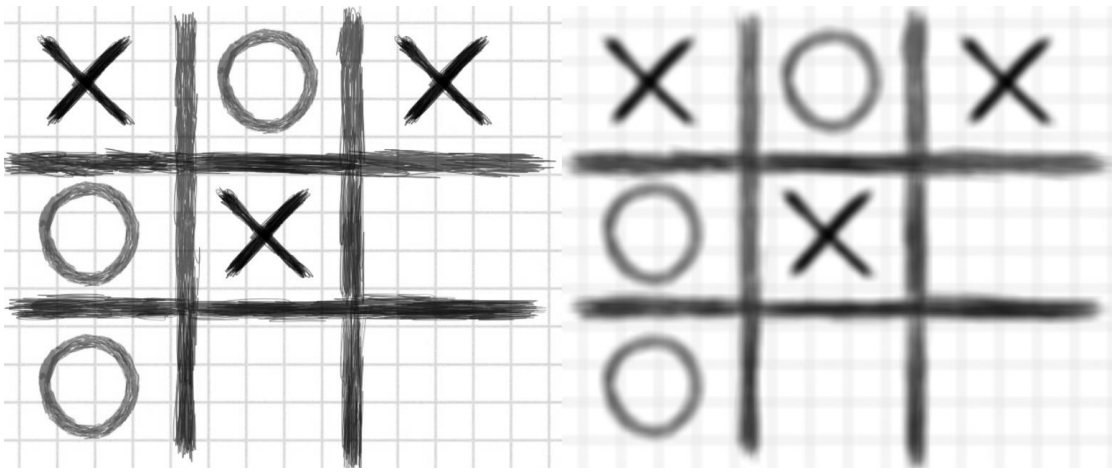


Figure 5.5: Gaussian Blur

As seen in Fig. 5.5 by reducing noise and smoothing the image, Gaussian blur can improve the accuracy of image recognition algorithms by making the objects features more distinguishable.

5.3.3 Adaptive Gaussian Thresholding

Thresholding is the process of separating objects or regions of interest from the background. It works by converting a greyscale image into a binary image, where pixels are classified as either foreground or background based on their intensity values. All pixels above a certain threshold are set to a high value (255 in our case), while all other pixels are set to a low value (0 in our case) according to the adaptive threshold formula:

$$dst(x,y) = \begin{cases} maxValue & if\ src(x,y) > T(x,y) \\ 0 & otherwise \end{cases}$$

where:

- $dst(x,y)$ represents the intensity value of pixel (x, y) at the output (destination) image
- $src(x,y)$ represents the intensity value of pixel (x, y) at the input (source) greyscale image
- *maxValue* represents the value 255 (white)
- $T(x,y)$ is a threshold calculated individually for the pixel at position (x, y)

Adaptive Gaussian was used over adaptive mean as it has better handling of gradients and illumination variations. This is especially important as most of our Tic-Tac-Toe games will be played on a white board. The application of adaptive Gaussian thresholding on our blurred greyscale image is shown in Fig. 5.6.

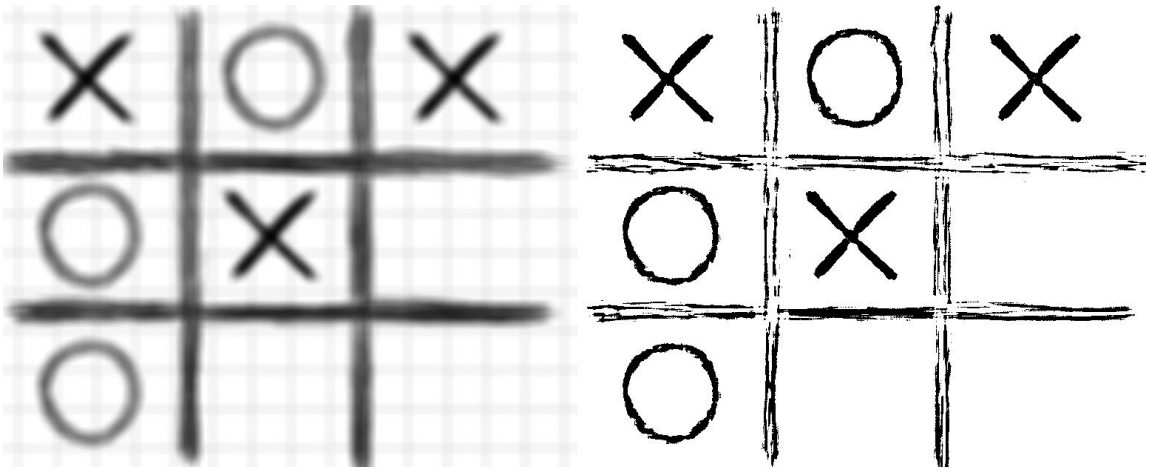


Figure 5.6: Adaptive Gaussian Thresholding

5.3.4 Edge Detection

Edge detection involves identifying boundaries between objects or regions of interest in an image. These edges represent significant changes in intensity or colour and provide essential information for tasks like object recognition, segmentation, and feature extraction. See Fig. 5.7.

The Laplacian operator is a widely used edge detection method that calculates the second derivative of the image intensity to detect edges. Its particularly effective at detecting edges where there is a sudden change in intensity, regardless of the direction of the change. Mathematically, the Laplacian operator is defined as the divergence of the gradient of the image:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

In practice, the Laplacian operator is implemented as a 3x3 convolutional kernel.

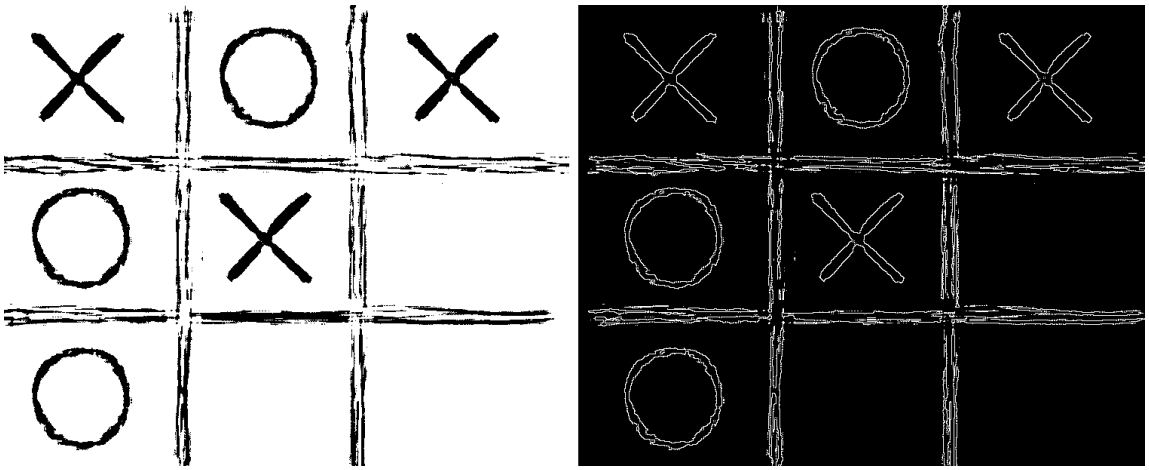


Figure 5.7: Laplacian Edge Detection

5.4 Board Detection

For NAO to be able to play the game, it needs to be able to tell where the game is being played. Detecting the board involves several image processing methods, such as corner detection, template matching and Hough Lines.

Corner Detection involves detecting the corners of the image using algorithms like the Harris corner detector [31] or the Shi-Tomasi detector [32]. Once the corners are detected, they can be used to estimate the geometry of the board, such as its size, orientation, and location. Once this is complete we can understand more about the image and move on to template matching.

Template matching is the process of matching predefined templates of the board with regions of the image to find instances of the board. This method works well when the appearance of the board is relatively consistent and can hand variations in scale, rotation, and perspective. The implementation of template matching was done using the following steps:

1. Template Creation: Initially, I created a dataset of templates with diverse features. Instead of relying on a single template image, I added multiple

templates representing various configurations and appearances of Tic-Tac-Toe boards (without any X's or O's). These templates serve as a reference pattern for comparison with regions of the input image.

2. Matching Process: The template matching algorithm then scans the input image, comparing the templates with different regions to identify potential matches. This comparison typically involves calculating the similarity between the template and each region using metrics like cross-correlation or mean squared error.

Although this method may be the easiest to implement, it can also be the least efficient as a small variation in the drawn board may lead to no detected boards or false positives as they may not "pass" the approval threshold for a template. In cases where no match was found Hough Lines was used.

Hough Lines uses Hough Transform [33] to detect straight lines that can be combined to form the boundaries of the board. This method is useful for detecting rectangular or square boards. See Fig. 5.8.

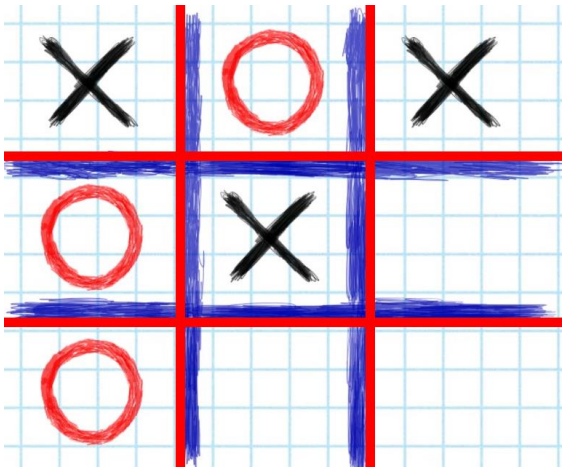


Figure 5.8: Hough Transform for board detection

5.5 Shape Detection (X's and O's)

5.5.1 Hough Transform for Circles

An efficient method of detecting simple shapes, such as lines, circles, or ellipses in images is to use Hough Transform (See Fig. 5.10). In the case of circle detection, the Hough Circle algorithm identifies circles based on the presence of edge points in the image.

The algorithm will first apply edge detection, using Laplacian edge detection. For each edge point in the image, a circle is parameterized by its centre coordinates (x_c, y_c) and its radius r. A 3D accumulator space is created with dimensions representing x_c, y_c, and r. The algorithm will then vote for potential circle centres in the accumulator space.

For the algorithm to correctly determine the circles, its parameters need to be refined as much as possible and the images given should be pre-processed to give the "cleanest" or least noisy image. A simple implementation of detecting circles can be seen in Fig. 5.9.

```
image = cv2.imread('image.jpg')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

blurred = cv2.GaussianBlur(gray, (9, 9), 10)

thresh = cv2.adaptiveThreshold(blurred, 255,
   cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,
   11, 2)

edges = cv2.Laplacian(thresh, cv2.CV_64F)
edges = cv2.convertScaleAbs(edges)


circles = cv2.HoughCircles(edges,
   cv2.HOUGH_GRADIENT,
   1, # Inverse ratio of the accumulator resolution
    20, # Minimum distance between the centers of the
   detected circles
```

```
    param1=55, # Higher threshold for the Canny edge
    detector

    param2=42, # Threshold for center detection

    minRadius=50, # Minimum radius to detect
    maxRadius=80 # Maximum radius to detect
)

# Draw the circles
if circles is not None:
     circles = circles[0]
     for circle in circles:
         x, y, r = map(int, circle)
         cv2.circle(edges, (x, y), r, (255, 255, 255),
    6)
         # Red dot in the center
         cv2.circle(edges, (x, y), 2, (255, 255, 255),
    5)
```

Figure 5.9: Implementation of Circle Detection (O Detection)

The output of the implementation in Fig. 5.9 can be seen in Fig. 5.11.

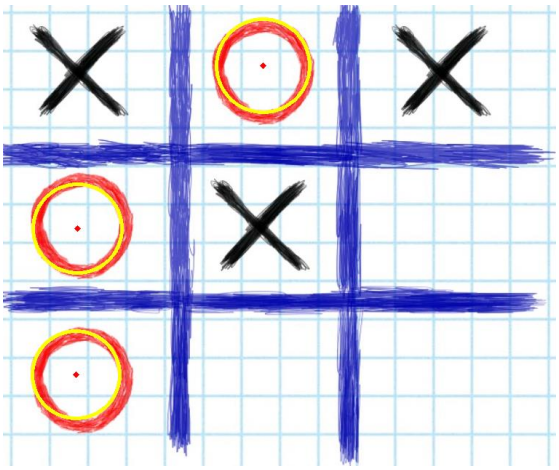


Figure 5.10: Circle Detection with hidden preprocessing

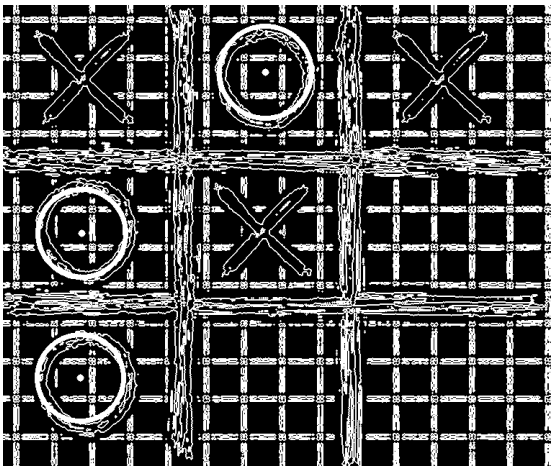


Figure 5.11: Circle Detection showing preprocessing

5.5.2 Template Matching for Crosses

As we saw in the previous section template matching is a fundamental technique use then computer vision for detecting predefined patterns or shapes within an image. In the context of cross detection template matching involves comparing a template image of the cross shape with the target image to find occurrences of that cross; this can be seen in Fig. 5.12.

As seen by the following code template matching is simple when implemented using OpenCV:

```
# Load the template for 'X'
template = cv2.imread('C:\\Users\\Mahdi\\Desktop\\Final
   Year Project\\nao_v4\\images\\x_template.png', 0)  #
   Replace 'x_template.png' with the path to your
   template image
template_h, template_w = template.shape[:2]

# Perform template matching
result = cv2.matchTemplate(gray, template,
   cv2.TM_CCOEFF_NORMED)

# Define a threshold for matches
threshold = 0.6

# Find locations where the correlation coefficient is
   above the threshold
loc = np.where(result >= threshold)
```

```
# Iterate through the locations and draw bounding boxes
   around matches
for pt in zip(*loc[::-1]):
    cv2.rectangle(image, pt, (pt[0] + template_w, pt[1]
   + template_h), (0, 255, 0), 2)
```

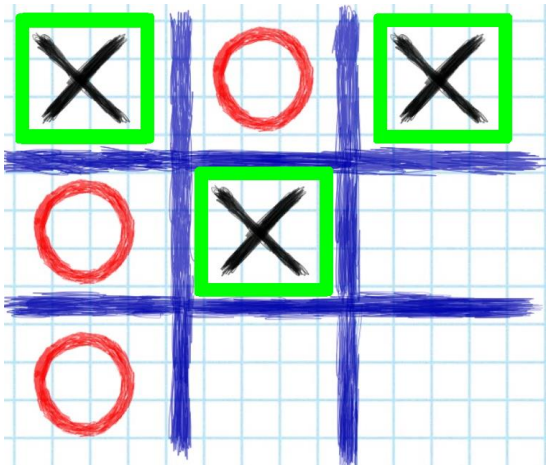


Figure 5.12: Template matching to detect ‘X’

Chapter 6 Gameplaying Module Implementation

6.1 Overview

Decision-making is a critical aspect of robotic autonomy, enabling robots to select actions that maximize their utility in achieving goals [16]. In the context of game-playing, robots employ specific strategies to make optimal moves. One prominent method is the Minimax algorithm. The Minimax algorithm models the game as a decision tree, where each node represents a game state, and the branches represent possible moves. The algorithm recursively evaluates the utility of each game state by assuming that both players will make optimal moves. This evaluation helps in identifying the move that maximizes the robot's chances of winning while minimizing the opponent's opportunities. In environments with large search spaces, enhancements like alpha-beta pruning can optimize the Minimax algorithm by reducing the number of nodes evaluated. These game-playing algorithms are crucial for enabling robots to navigate the complexities of strategic interactions and make intelligent decisions in competitive scenarios.

6.2 Game Theory

Game theory is a branch of mathematics that involves analysing strategic interactions among rational decision-makers, offering insights into how entities make choices when their interests are opposed. This field provides a structured approach to understanding the dynamics of decision-making in situations where the outcomes depend not only on one's actions but also on the actions of others [18].

Simply, game theory serves as a toolkit for studying scenarios where multiple actors, each with their own objectives, must make decisions while considering the potential

responses of others. It helps uncover patterns of behaviour, predict outcomes, and identify optimal strategies in various competitive or cooperative settings.

Expanding on its significance, game theory finds applications in a wide range of fields, including economics, political science, biology, and computer science. It enables us to understand phenomena such as market competition, negotiations, conflict resolution, evolutionary dynamics, and strategic interactions in social networks [17].

Adversarial game playing is a fundamental concept within game theory, particularly in scenarios where there are two or more players with conflicting interests. One of the most common and simplest examples of this is a two-player, zero-sum game [34], where the total gains and losses among the players sum to zero, meaning that any gain for one player corresponds to an equivalent loss for the other.

Tic-tac-toe is a classic example of a two-player, zero-sum game. In this game, two players take turns marking spaces in a 3x3 grid with their respective symbols, usually "X" and "O", with the objective of creating a row, column, or diagonal of their symbols. If one player succeeds in forming a complete line of their symbols before the grid is filled, they win; otherwise, if the grid fills up without a complete line being formed, the game is a draw.

What makes tic-tac-toe a zero-sum game is that each player's gain (a win) corresponds directly to the other player's loss. If Player A wins, Player B loses, and vice versa. Additionally, the total number of possible outcomes in tic-tac-toe is limited, making it a game of perfect information [35], where both players have full knowledge of all previous moves and possible future moves.

6.3 Minimax Algorithm

6.3.1 Understanding Minimax

Minimax is a decision-making algorithm used in game theory that aims to minimize the potential loss in a worst-case scenario. Minimax requires perfect information for it to be effective as seen in Fig. 6.1. It is also commonly applied to 2-player games, where the value of the gain of one player is the same as the value of the loss of the other. The idea behind minimax is for a player to follow a strategy that maximizes their minimum possible payoff while assuming their opponent will play optimally [18].

Looking at Fig. 6.1 we can see a general look of a Tic-Tac-Toe game tree and what the individual states are defined as. Minimax works like this, If the opponent is the maximizing player, our aim is to minimize our score. We wait for MAX to play their move and from there we will evaluate every leaf node using an evaluation function, which is the utility in the terminal state that estimates the "desirability" of that game state for the current player. An example evaluation function assigns values +1 for the maximizing player's win, 0 for a draw, and -1 for the minimizing player's win. For a 2-ply tree, the Minimax algorithm evaluates the tree up to two moves ahead, alternating between the MAX and MIN players at each level.

An example of a 2-ply game tree is seen in Fig. 6.2. In the first ply, MAX makes a move. In the second ply, MIN responds to each of MAX's possible moves. At the leaf nodes of this 2-ply tree, the evaluation function assigns scores based on the desirability of the game state from the perspective of the current player (MIN). The algorithm then backtracks, propagating these scores up the tree. At the MIN level, the algorithm chooses the minimum score from the leaf nodes, as MIN aims to minimize the score. Once the entire 2-ply tree has been evaluated, the algorithm will select the move that

leads to the best score for MIN, ensuring that the response to MAX's initial move is optimal from MIN's perspective.

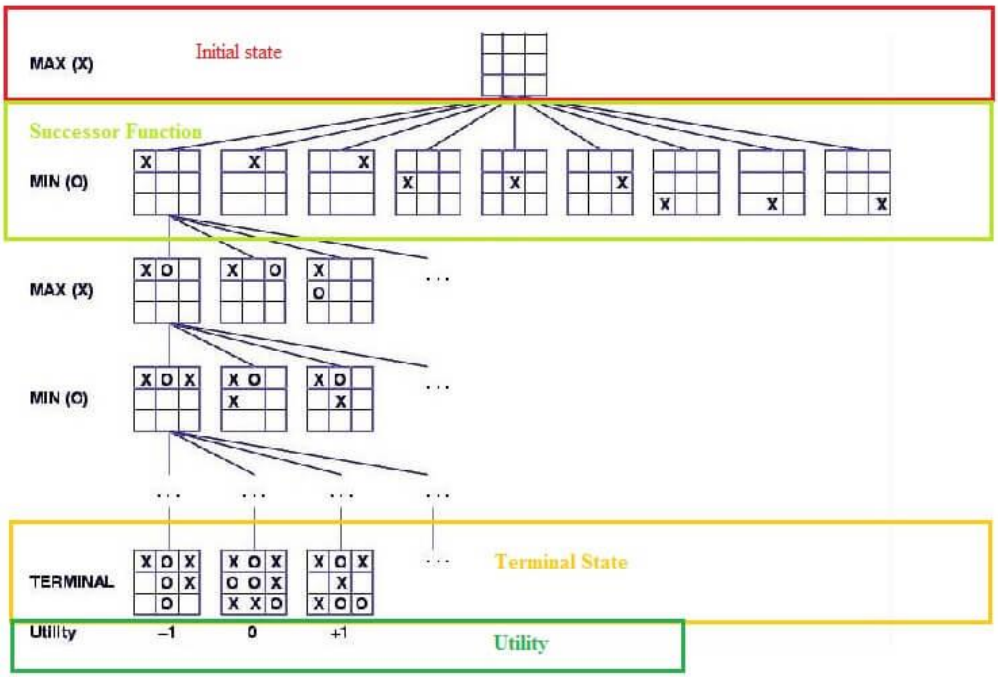


Figure 6.1: Minimax for Tic-Tac-Toe

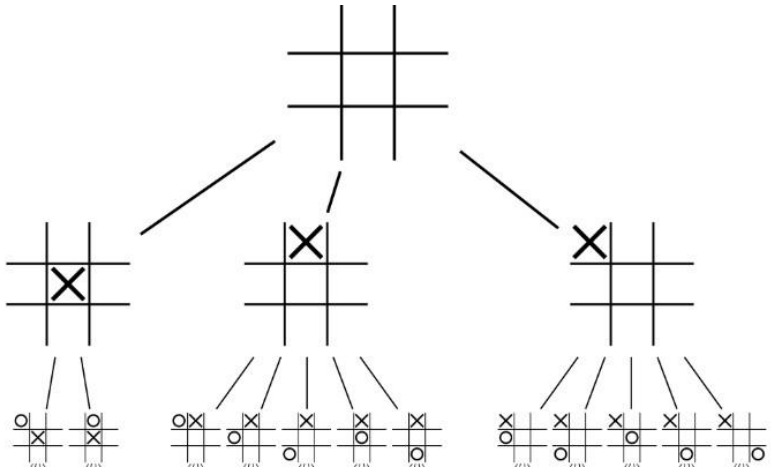


Figure 6.2: 2-ply Tic-Tac-Toe Game Tree

Even though minimax can be used in many different styles of gameplay, it is most effective in board games, such as Chess, Tic-Tac-Toe, Othello, and Checkers.

6.3.2 Alpha-Beta Pruning

In game trees the number of possible game states grows exponentially with the depth of the tree. Even though we cannot eliminate this exponential growth we can significantly reduce it using optimisation techniques such as Alpha-Beta pruning. Using Alpha-Beta pruning we cut of large parts of the tree that don't affect our outcome [18].

In Alpha-Beta Pruning we keep track of two parameters, α and β, which are used to keep track of the best choices found so far. A represents the highest value found for the MAX player, while β represents the lowest value found for the MIN player.

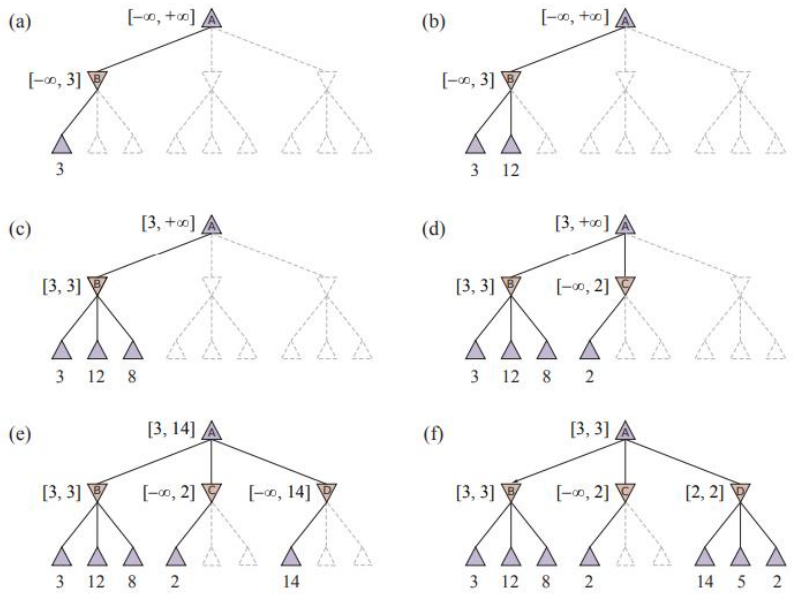


Figure 6.3: Alpha-Beta Pruning [18]

In Fig. 6.3, we can see how Alpha-Beta pruning works. As the search progresses and the nodes are evaluated, we keep track of the two parameters. When evaluating a MAX node, if α is updated to a value greater than or equal to β, it means that the MAX player has found a move that is at least just as good as a move previously considered by MIN.

Therefore, we no longer need to continue as MIN would never choose this path, hence pruning occurs, this can be seen in Fig. 6.2 part (d) [18].

Oppositely when evaluating a MIN node, if β is updated to a value less than or equal to α, it means the MIN player has found a move that is at least just as good a move previously considered by MAX. Therefore, pruning occurs as MAX would never choose that path [18].

6.4 Implementing Minimax for Tic-Tac-Toe

When implementing game-playing algorithms such as minimax, representing the game state effectively is critical. Game states contain all the necessary information and misrepresenting them can mean the algorithm does not function properly or may take in wrong information.

Implementing game states for board games is very simple as we have 2D grids. These grids can easily be represented as either n x n arrays or matrices. In the case of Tic Tac Toe, we can use a simple 3x3 array as they are fast, easy to manipulate and easy to read from. Each player will be represented by either an ‘X’ or ‘O’.

After we have implemented the game states, we need to be able to evaluate each game state including winning positions, blocking moves, centre control and corner control. Minimax will allow us to explore all possible moves recursively, since we will be scoring each move, we can fulfil all the above requirements. We can see the pseudocode of how it was implemented below:

```
function minimax(node, depth, maximizingPlayer):
    if depth = 0 or node is a terminal node:
        return the heuristic value of node

    if maximizingPlayer:
        bestValue := -∞
        for each child of node:
            value := minimax(child, depth - 1, FALSE)
```

```
            bestValue := max(bestValue, value)
        return bestValue

    else:  // minimizing player
        bestValue := +∞
        for each child of node:
            value := minimax(child, depth - 1, TRUE)
            bestValue := min(bestValue, value)
        return bestValue
```

In the implementation of the Minimax algorithm for the tic-tac-toe game, a maximum depth of 9-ply has been utilized. This ensures that the algorithm considers every possible move sequence up to the completion of the game, where each ply represents a single move by one player. By exploring the full depth of the game tree, the algorithm guarantees that it evaluates all potential outcomes, thereby ensuring optimal move selection. This comprehensive look ahead ensures that the algorithm is robust and reliable, as it exhaustively examines every possible game state, preventing any scenario where the algorithm might fail to function correctly due to an incomplete evaluation of possible moves.

6.4.1 Evaluation Function

Creating an effective evaluation function for Tic-Tac-Toe involves designing a heuristic that can estimate the desirability of a given board state. While the simplest evaluation function assigns a value of +1 for a win for the maximizing player (X), -1

for a win for the minimizing player (O), and 0 for a draw or ongoing game, a more detailed function can help the Minimax algorithm make better decisions in non-terminal states [17]. Here is the evaluation function used for this implementation:

Winning state:

• +10 if the maximizing player (X) wins.

• -10 if the minimizing player (O) wins.

Draw state:

• 0 for a draw.

Intermediate states:

Evaluate based on potential future wins:

• +3 for each row, column, or diagonal where the maximizing player (X) can still win.

• -3 for each row, column, or diagonal where the minimizing player (O) can still win.

Evaluate based on immediate opportunities:

• +1 for each row, column, or diagonal where the maximizing player (X) has two in a row and an empty space.

• -1 for each row, column, or diagonal where the minimizing player (O) has two in a row and an empty space.

Essentially, the worst possible score for each respective player is initially assumed, and any improvements are sought. This is a recursive function that will go through every possible move from the current game state and find the next best move, this may be a block or an attempt to get 3 in row.

Even though the above evaluation function is more intricate, my look-ahead reaches the end of the game tree, a straightforward utility function was used to evaluate the outcomes. Specifically, a utility value of +1 was assigned for a win for the maximizing player, -1 for a win for the minimizing player, and 0 for a draw. This approach ensures

clear and unambiguous evaluation of terminal states, allowing the Minimax algorithm to make precise decisions based on the definitive outcomes of win, loss, or draw. While this basic utility function is effective for evaluating terminal states, it is important to note that more complex evaluation functions could be developed to enhance decision-making in non-terminal states by estimating the desirability of a given board configuration more accurately [17]. However, for the purposes of this implementation, the primary focus was on reaching terminal states, thereby justifying the use of these specific utility values.

Chapter 7 Results and Analysis

7.1 Introduction

NAO demonstrated significant success in playing Tic-Tac-Toe against multiple human opponents, showcasing its advanced interactive and decision-making capabilities. Through extensive gameplay sessions, the NAO robot consistently made optimal moves, effectively challenging its human counterparts and illustrating the robustness of its integrated artificial intelligence and computer vision systems.

The NAO robot's success in playing Tic-Tac-Toe against multiple opponents can be attributed to the integration of computer vision algorithms and the Minimax algorithm, allowing it to accurately perceive the game board and make strategic decisions. Through extensive testing and evaluation, the robot's ability to interpret visual inputs, recognize game states, and respond accordingly was rigorously assessed and improved. These tests highlighted its proficiency in object recognition, move prediction, and interaction based on visual cues. Preliminary results confirmed the successful implementation of these algorithms, enabling the NAO robot to accurately detect and respond to predefined visual stimuli. Despite challenges related to real-time processing and adaptation to dynamic environments, iterative testing and refinements enhanced the system's reliability and performance. Ultimately, the outcomes demonstrated the NAO robot's potential for engaging and interactive gameplay, underscoring its effectiveness as a competent and interactive player in Tic-Tac-Toe.

7.2 Results

During the extensive testing phase, where NAO engaged in playing Tic-Tac-Toe with many human players, valuable insights were gained, revealing previously overlooked bugs. One significant observation was the impact of varying light conditions on NAO's

performance. It became apparent that differentiation in lighting could hinder the robot's detection capabilities, leading to inaccuracies and potential errors in gameplay. Additionally, an unexpected challenge emerged as NAO's operational duration increased; after approximately 30 minutes of continuous activity, the robot's internal temperature rose, affecting its performance and eventually causing NAO to enter "sleep mode". Furthermore, the interaction between human players and the robot introduced complexities, particularly concerning hand movements. Instances were noted where players' hands obstructed the robot's field of view, resulting in false detections and gameplay errors. These findings underlined the importance of robustness testing in real-world scenarios, enabling the identification and resolution of issues that may not manifest under controlled laboratory conditions.

7.2.1 Object Recognition

Initially, NAO was successfully tested and calibrated to detect a single object within the Tic Tac Toe environment using image processing techniques and object recognition algorithms. Through numerous experiments, the system's performance was evaluated under various lighting conditions, ranging from ideal to increasingly challenging scenarios. These experiments facilitated the fine-tuning of parameters to optimize detection accuracy and reliability. Additionally, adjustments were made to the camera's distance and angles to enhance the system's robustness across diverse conditions.

7.2.2 Grid Detection

The evaluation of the NAO robot's ability to recognize individual objects within the Tic-Tac-Toe game environment revealed significant challenges, particularly concerning the detection of the grid. The potential misclassification of elements posed obstacles to accurate detection, especially at intersections with game pieces. Algorithms such as the Hough Transform were employed to identify the lines forming the

boundaries of each cell within the grid. However, noise and visual artifacts complicated this process, leading to the detection of redundant or undesired lines (see Fig. 7.1).

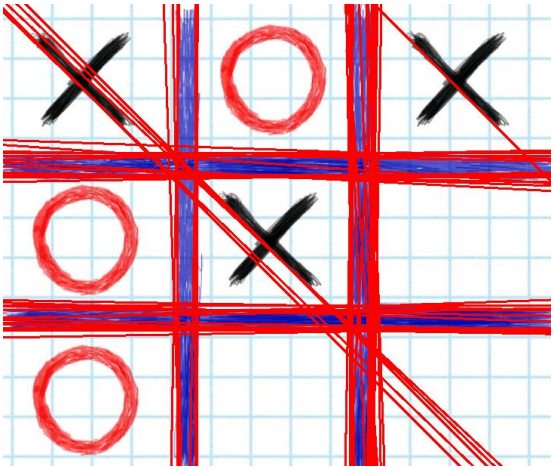


Figure 7.1: Grid misclassification

7.2.3 Circle (O) Detection

Circle detection using the Hough Transform was a straightforward process that requires minimal effort when implementing. By fine-tuning parameters such as threshold, radius, and resolution, the algorithm can be adapted to different drawing styles and environments, making it a versatile tool for circle detection in computer vision applications.

7.2.4 Cross (X) Detection

Creating a robust system for detecting crosses or "X" shapes using template matching hinges on assembling a diverse dataset that captures the variability in how individuals draw crosses across different environments and surfaces. A diverse dataset is crucial due to the myriad styles and environments in which crosses are drawn. Variations in line thickness, angle, and drawing surface introduce complexities that necessitate robust training data. Without such diversity, the model may struggle to generalize to unseen data or accurately detect crosses in real-world scenarios.

Various strategies can be employed to build a diverse dataset. Crowdsourcing cross drawings from individuals across different demographics and environments can capture diverse drawing styles effectively. Simulated environments can generate synthetic data to augment the dataset, covering a broad spectrum of lighting conditions and backgrounds. Additionally, capturing images of crosses drawn in various real-world environments ensures variability in surface types and environmental conditions.

By assembling a diverse dataset and training the template matching algorithm on annotated images, significant improvements in object detection and classification accuracy can be achieved. The resultant system becomes more robust to variations in drawing styles, environments, and surfaces, facilitating accurate cross detection in real-world scenarios. The successful outcome of these efforts is exemplified in Fig. 7.2, showcasing a clear and accurately recognized Tic Tac Toe board with correctly identified and positioned game pieces.

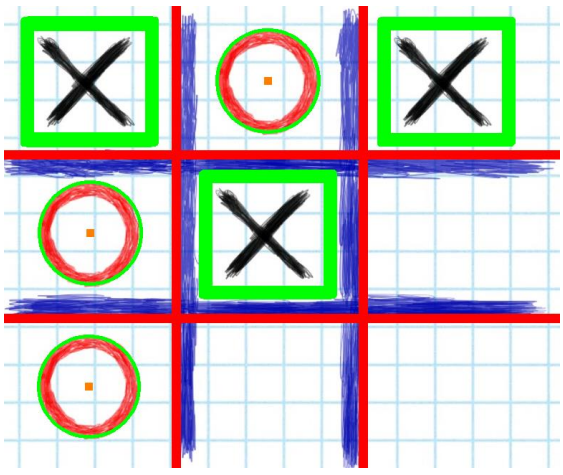


Figure 7.2: Full detection of game board

7.2.5 Minimax

In this project, two distinct strategies were employed when implementing the minimax algorithm for Tic-Tac-Toe. The first approach utilized a 2-ply lookahead mechanism coupled with an evaluation function to assess each non-terminal state. The second strategy opted for a full-depth search, employing alpha-beta pruning to traverse the game tree until reaching terminal states. Upon conducting empirical evaluations using various game scenarios and move sequences in Tic-Tac-Toe, it was observed that the evaluation function introduced a computational overhead, resulting in a time delay of approximately 500 milliseconds compared to the full-depth search, this can be seen in Fig. 7.3. Consequently, the decision was made to proceed with the deployment of the full-depth search approach due to its superior computational efficiency and ability to exhaustively explore the game space. These findings underscore the importance of selecting appropriate search algorithms and evaluation functions tailored to the characteristics of the problem domain, such as Tic Tac Toe, for optimal performance in game playing scenarios.

```
Running MINIMAX 2-PLY EVALUATION FUNCTION...
Move time: 534 ms
Move: 1 1
Move time: 619 ms
Move: 2 0
Move time: 813 ms
Move: 0 2
Move time: 971 ms
Move: 0 1
Move time: 489 ms
Move: 1 2
Move time: 996 ms
Move: 0 2
Move time: 966 ms
Move: 2 0
Move time: 935 ms
Move: 2 2
Move time: 856 ms
Move: 1 0
Move time: 697 ms
Move: 2 1
Average execution time for MINIMAX 2-PLY W/ EVALUATION FUNCTION: 787 ms
Running MINIMAX FULL-DEPTH ALPHA BETA...
Move time: 230 ms
Move: 1 2
Move time: 318 ms
Move: 0 0
Move time: 222 ms
Move: 1 2
Move time: 385 ms
Move: 1 0
Move time: 349 ms
Move: 2 2
Move time: 222 ms
Move: 2 0
Move time: 318 ms
Move: 1 1
Move time: 334 ms
Move: 1 1
Move time: 328 ms
Move: 0 1
Move time: 364 ms
Move: 1 2
Average execution time for MINIMAX W/ FULL-DEPTH + ALPHA BETA: 307 ms
MINIMAX FULL-DEPTH ALPHA BETA optimized by approximately 480 ms
```

Figure 7.3: Evaluation of Minimax 2-ply vs Full-Depth Search with Alpha-Beta

7.3 Challenges

7.3.1 Object Recognition

The process of testing and calibrating NAO presented several challenges. Fine-tuning parameters for object detection required extensive experimentation and iterative adjustments, consuming significant time and resources. Moreover, evaluating the system's performance under different lighting conditions posed a challenge, as variations in illumination levels could affect detection accuracy. Additionally, optimizing the camera's distance and angles to ensure robust operation across varied environments necessitated thorough testing and validation. Overall, addressing these challenges was essential to develop a reliable and versatile object detection system within the Tic Tac Toe environment.

7.3.2 Grid Detection

Detecting the Tic-Tac-Toe grid entailed addressing complexities inherent in its structure. The presence of four lines delineating each cell required accurate identification to segment the grid effectively. The use of algorithms like the Hough Transform was hindered by noise and other visual disturbances, resulting in the detection of extraneous lines and potential misclassification of game pieces as lines. To mitigate these challenges, a combination of image processing techniques and custom filters was employed. These filters aimed to differentiate between grid lines and game pieces, enhancing the accuracy of board layout recognition. Despite these efforts, challenges persisted, particularly when attempting to detect boards drawn on lined paper. The background "grids" of the paper often confounded the algorithm, leading to errors in object recognition.

To address the misclassification of background grids, various preprocessing techniques were applied, notably Gaussian blur, Laplacian edge detection, and adaptive

thresholding. Gaussian blur was instrumental in reducing image noise and enhancing object boundaries' clarity, thus facilitating subsequent processing steps. Laplacian adaptive thresholding played a crucial role in segmenting the image into distinct regions, particularly in distinguishing grid lines from game pieces. By dynamically adapting the threshold based on local image properties, this technique effectively highlighted grid edges, aiding in their extraction from the background. With these preprocessing techniques implemented, the object recognition algorithm exhibited improved accuracy and reliability, even in challenging scenarios such as boards drawn on lined paper or under varying lighting conditions.

7.3.3 Cross Detection

The primary challenge lies in curating a dataset that adequately represents the variability in cross drawings. This entails sourcing cross drawings from diverse individuals and environments to cover a wide range of styles, lighting conditions, backgrounds, and surfaces. Crowdsourcing and simulation techniques can aid in diversifying the dataset, but ensuring comprehensive coverage remains a challenge. Annotating each image with cross location and orientation is essential for training the template matching algorithm accurately. However, this process can be labour-intensive and may require careful consideration of metadata such as drawing style and environmental conditions.

7.4 Latency Issues and Optimisations

Latency issues such as data transmission delays from the cameras to the processing unit can impact the robot's ability to recognize the game board quickly. Optimizations such as reducing data processing overhead or optimizing data transfer rates can mitigate this issue. Additionally, complex algorithms for object recognition and decision-making

may introduce latency. Streamlining these algorithms and leveraging parallel processing can improve real-time performance.

7.4.1 Data Processing Optimisations

Reducing the data overhead was crucial for minimising the latency of the robot. This was achieved by processing as much of the information on NAO, and any other data too large to process was compressed and sent to the computer.

7.4.2 Algorithmic Optimisation

Designing algorithms that require lower computational complexity can reduce processing time significantly. Simplifying the models, optimising data structures, and employing heuristic approaches were all necessary to reduce the latency.

7.4.3 Conclusion

Real-time processing is crucial for the NAO robot's performance in playing Tic-Tac-Toe. By integrating sensor data, optimizing algorithms, and mitigating latency issues, the robot can effectively engage in real-time gameplay, providing an enjoyable and interactive experience for users.

Chapter 8 Summary and Future Work

8.1 Summary

In this thesis project, the development and implementation of a NAO humanoid robot designed to play Tic-Tac-Toe against a human opponent were explored. The primary objective was to integrate computer vision and artificial intelligence to enhance the robot's interactive capabilities. Leveraging the OpenCV library, the project enabled the NAO robot to perceive and interpret its environment, focusing on computer vision and making informed decisions based on visual stimuli.

The methodology included implementing computer vision algorithms within the OpenCV framework to process visual data captured by the NAO robot's cameras. This allowed the robot to recognize the Tic-Tac-Toe board and the moves made by the human player. Additionally, the project incorporated the minimax algorithm to enable the robot to make optimal moves during gameplay. The NAOqi SDK and Python programming were utilized to ensure seamless integration with the robot's hardware and software architecture, facilitating effective human-robot interaction.

Extensive testing and evaluation demonstrated the system's ability to accurately identify and respond to game states, showcasing the NAO robot's potential for engaging and interactive gameplay. The initial testing scenarios included the NAO robot's visual perception capabilities, such as object recognition, facial detection, and interaction based on visual cues. Preliminary results indicated successful implementation of computer vision algorithms, allowing the robot to accurately identify and respond to predefined visual stimuli, including playing Tic-Tac-Toe.

However, challenges related to real-time processing and adapting to dynamic environments were acknowledged and discussed. The project's outcome contributes to

the field of robotics by showcasing practical applications of computer vision and AI on the NAO robot, enhancing its perceptual capabilities, and enabling more sophisticated human-robot interactions.

Future work may address optimization challenges, expand the range of visual tasks, and explore advanced AI techniques for complex decision-making scenarios. Potential areas for further exploration include improving the robot's adaptability to dynamic environments, enhancing real-time processing capabilities, and integrating more advanced AI algorithms to handle more complex interactions. This project opens avenues for further exploration at the intersection of robotics, computer vision, and artificial intelligence, promising exciting advancements in the field.

8.2 Future Directions

8.2.1 System Refinements

There are many improvements that can be made to the current Tic-Tac-Toe game played by NAO, primarily in adding drawing capabilities. But moreover, much more can be added such as gesture tracking, sentiment analysis, facial tracking, and streamlines conversation.

8.2.1.1 Drawing Capabilities

Integrating drawing capabilities into the NAO robot's communication system presents a novel way of conveying information beyond speech. This enhancement would involve equipping the robot with a drawing tool and developing algorithms to convert text or concepts into drawings. This would allow the human player to feel more immersed in the game.

Additionally, drawing could be applied to other tasks and serve as an effective means for the robot to illustrate complex ideas, instructions, or responses. For instance, in

educational settings, NAO could visually explain concepts like geometry or anatomy, enhancing understanding and engagement among students. This feature would require advancements software components, including mechanisms for precise movement control and algorithms that can translate textual inputs into coherent visual representations.

8.2.1.2 Streamlined Conversations, Facial Tracking, and Gesture Tracking:

Refining the system for streamlined conversations, accurate facial tracking, and precise gesture analysis is essential for improving the fluidity and effectiveness of communication with the NAO robot. Currently, the robot's ability to engage in natural conversations, accurately track facial expressions, and interpret gestures is limited. Advancements in natural language processing algorithms can enable the robot to understand and respond to users' queries and commands more effectively. Furthermore, improvements in facial tracking technology would allow the robot to recognize subtle facial expressions, enhancing its ability to gauge users' emotions and tailor its responses accordingly. Additionally, enhancing gesture analysis would enable the robot to interpret users' gestures more accurately, facilitating more intuitive and interactive communication experiences. These refinements would require advancements in machine learning and computer vision techniques, as well as optimization of the robot's hardware components to support real-time processing of complex inputs.

8.2.2 Expanding Applications

Expanding the applications of NAO would involve incorporating advanced computer vision and AI capabilities to facilitate new use cases.

8.2.2.1 Multi-Game Capabilities

Expanding the number of games NAO can play enhances its versatility and engagement potential across various domains. By developing a diverse library of games with various

modes and difficulty levels, the robot can cater to different age groups and interests. Moreover, enabling the robot to adjust its position and posture dynamically according to the requirements of each game enhances the immersive gaming experience. For example, in educational settings, the robot could play interactive learning games that adapt to students' skill levels and preferences, fostering personalized and effective learning experiences. Implementing multi-game capability requires not only the development of new game software but also enhancements in the robot's motion control and coordination capabilities to ensure smooth transitions between different game modes.

8.2.2.2 Advanced Computer Vision and AI Capabilities

Integrating advanced computer vision and AI capabilities into the NAO robot opens new possibilities for its deployment in diverse applications. Currently, the robot's perception and understanding of its environment are limited, this is mainly due to the low-resolution cameras NAO is equipped with and the low-quality sonar. Object recognition technology would allow the robot to identify and interact with objects in its surroundings, enabling it to perform tasks such as fetching objects or assisting with household chores. Furthermore, scene understanding capabilities would enhance the robot's ability to navigate complex environments and interact with users contextually. For instance, in retail settings, the robot could guide customers to specific products, provide information about promotions, and even assist with checkout processes, enhancing the overall shopping experience. Additionally, emotion recognition technology would enable the robot to recognize and respond to users' emotional states, fostering more empathetic and personalized interactions. These advancements necessitate the integration of state-of-the-art algorithms and sensors, as well as

enhancements in the robot's processing power and memory capacity to handle the increased computational demands.

8.3 Final Thoughts

The NAO robot indeed possesses remarkable capabilities, but utilising its full potential requires dedicated time and effort, especially during the initial stages of programming. Developing the initial "skeleton" of the program lays the foundation for the robot's functionality, but it often involves extensive research, coding, and testing to ensure that the robot can perform its intended tasks effectively.

In my experience with the NAO robot project, creating this initial skeleton demanded a deep understanding of the robot's hardware and software architecture, as well as proficiency in programming languages such as Python. I needed to familiarize myself with the robot's capabilities, including its sensors, actuators, and communication interfaces, to design a program that could effectively utilize these resources.

Once the skeleton of the program was in place, the real power of the NAO robot became apparent. With this framework established, adapting the robot to different tasks became significantly easier, but still very demanding. I could leverage existing code and modules, making modifications and additions as necessary to tailor the robot's behaviour to specific applications.

For example, if I initially programmed the robot to perform a simple task like recognizing and greeting people, I could easily extend its capabilities to include additional functionalities such as responding to voice commands, providing information about nearby objects, or even dancing to music. By reusing and building upon the existing codebase, I could rapidly iterate on different ideas and prototypes, saving time and effort in the development process.

Throughout the project, I gained invaluable insights into how robots work and how they communicate with their software. I learned about the intricacies of robotic perception, cognition, and interaction, as well as the challenges and limitations inherent in designing robotic systems. Understanding the underlying principles of robotics and software development enabled me to troubleshoot issues effectively and optimize the robot's performance.

Moreover, working with the NAO robot provided me with hands-on experience in robotics research and development, preparing me for future endeavours in this field. I gained practical skills in programming, problem-solving, and project management, as well as a deeper appreciation for the potential impact of robotics on various industries and domains.

In conclusion, while the initial development of the NAO robot program may require significant time and effort, the payoff in terms of versatility, adaptability, and learning experience is well worth it. By investing in the foundational framework and leveraging it for different tasks, I was able to maximize the robot's capabilities and expand my own knowledge and skills in robotics and software engineering.

References

[1] Inc, S. R. A. (n.d.). NAO. https://us.softbankrobotics.com/nao

[2] Accuray | CyberKnife. (2024, March 22). Home - CyberKnife. Retrieved from CyberKnife: https://cyberknife.com/

[3] Aida Amirova, 1. N., Yadollahi, E., Sandygulova, A., & Johal, W. (2021, November 19). 10 Years of Human-NAO Interaction Research: A Scoping Review. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8640132/

[4] Bäckström, T. (n.d.). Concatenative speech synthesis — Introduction to Speech Processing. Retrieved from Aalto: https://speechprocessingbook.aalto.fi/Synthesis/Concatenative_speech_synthesis.html

[5] Bonarini, A. (2020). Communication in Human-Robot interaction. Current Robotics Reports, 279-285.

[6] Brownlee, J. (2021, January 26). A gentle introduction to object recognition with deep learning. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/object-recognition-with-deep-learning/

[7] Decision-Making and Control under Uncertainties for Robotic and Autonomous Systems. (2024). Retrieved from MDPI: https://www.mdpi.com/journal/robotics/special_issues/Decision-Making

[8] Fagella, D. (2020, March 14). Machine Learning in Robotics – 5 Modern Applications. Retrieved from Emerj Artificial Intelligence Research: https://emerj.com/ai-sector-overviews/machine-learning-in-robotics/

[9] Henrich, J. (2023, January 26). Object Recognition: an introduction and overview (with Tensorflow). Retrieved from Tensor Science: https://www.tensorscience.com/posts/object-recognition-an-introduction-and-overview.html

[10] RobotsGuide (n.d.). Retrieved from https://robotsguide.com/robots/nao/

[11] King, S. (2011). An introduction to statistical parametric speech synthesis. Sadhana/Sādhanā, 837-852.

[12] Kundu, R. (2024, April 10). Image Processing: Techniques, Types, & Applications [2023]. Retrieved from V7: https://www.v7labs.com/blog/image-processing-guide

[13] Li, C., & Wang, X. (2016). Visual localization and object tracking for the NAO robot in dynamic environment. Retrieved from https://ieeexplore.ieee.org/abstract/document/7831973

[14] Liu, Q., Zhang, C., Song, Y., & Pang, B. (2018). Real-Time Object Recognition Based on NAO Humanoid Robot. Retrieved from https://ieeexplore.ieee.org/document/8726687

[15] Macaulay, M., & Shafiee, M. (2022). Machine learning techniques for robotic and autonomous inspection of mechanical systems and civil infrastructure. Autonomous intelligent systems. doi:10.1007/s43684-022-00025-3

[16] Marcos Maroto-Gómez, F. A.-M.-G. (2023). A Systematic literature review of Decision-Making and Control Systems for autonomous and social Robots. International journal of social robotics, 745. doi:10.1007/s12369-023-00977-3

[17] Marwala, T. (2023). Game Theory in Politics. In T. Marwala, . Game Theory in Politics. In: Artificial Intelligence, Game Theory and Mechanism Design in Politics. Singapore: Palgrave Macmillan.

[18] Norvig, P. (2021). A Modern Approach, Global Edition. In S. R. Norvig, ARTIFICIAL INTELLIGENCE. Createspace Independent Publishing Platform.

[19] Guggemos, J., Seufert, S., Sonderegger, S., & Burkhard, M. (2022). Social Robots in Education: Conceptual Overview and Case Study of use. In Cognition and exploratory learning in the digital age (pp. 173–195). https://doi.org/10.1007/978-3-030-90944-4_10

[20] OpenCV AI Kit course. (n.d.). Retrieved from https://roboflow.com/course

[21] Learning Computer Vision using a Humanoid Robot. (2019, April 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8725196

[22] Patel, M. (2023, October 23). The complete guide to image preprocessing techniques in Python. Retrieved from Medium: https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c

[23] Rabb, N., Law, T., Chita-Tegmark, M., & Scheutz, M. (2021, July 2021). An Attachment Framework for Human-Robot Interaction. Retrieved from https://link.springer.com/article/10.1007/s12369-021-00802-9

[24] Robaczewski, A., Bouchard, J., Bouchard, K., & Gaboury, S. (2020). Socially assistive robots: the specific case of the NAO. International journal of social robotics, 795-831. doi:10.1007/s12369-020-00664-7

[25] Snyder, W., & Qi, H. (2017). Fundamentals of Computer Vision. North Carolina: Cambridge University Press. doi:10.1017/9781316882641

[26] Tam, A. (2024, January 22). Machine Learning in OpenCV (7-Day Mini-Course). Retrieved from MachineLearningMastery.com: https://machinelearningmastery.com/machine-learning-in-opencv-7-day-mini-course/

[27] Tanevska, A., Rea, F., Sandini, G., Cañamero, L., & Sciutti, A. (2020, October 19). A Socially Adaptable Framework for Human-Robot Interaction. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7806058/

[28] Antonietti, A., Martina, D., Casellato, C., D'Angelo, E., & Pedrocchi, A. (2019). Control of a humanoid NAO robot by an adaptive bioinspired cerebellar module in 3D motion tasks. Computational Intelligence and Neuroscience, 2019, 1–15. https://doi.org/10.1155/2019/4862157

[29] Banaeian, H., & Gilanlioglu, I. (2021b). Influence of the NAO robot as a teaching assistant on university students' vocabulary learning and attitudes. Australasian Journal of Educational Technology, 71–87. https://doi.org/10.14742/ajet.6130

[30] Wikipedia contributors. (2024, March 18). Nao (robot). Wikipedia. https://en.wikipedia.org/wiki/Nao_%28robot%29

[31] Science, B. O. C., & Science, B. O. C. (2024, March 18). Harris Corner Detection explained | Baeldung on computer science. Baeldung on Computer Science. https://www.baeldung.com/cs/harris-corner-detection

[32] Gandhi, N. (2018, August 16). Harris corner detection and Shi-Tomasi corner detection. Medium. https://medium.com/pixel-wise/detect-those-corners-aba0f034078b

[33] Cantoni, V., & Mattia, E. (2013). Hough transform. In Springer eBooks (pp. 917–918). https://doi.org/10.1007/978-1-4419-9863-7_1310

[34] Bacharach, M. (1987). Zero-Sum Games. In Palgrave Macmillan UK eBooks (pp. 1–4). https://doi.org/10.1057/978-1-349-95121-5_1658-1

[35] Osborne, M., & Rubinstein, A. (1995). A course in game theory. https://www.semanticscholar.org/paper/A-Course-in-Game-Theory-Osborne-Rubinstein/ef336fe9c04559654936413f4910a54b7ae5028c

[36] Sharma, N., & Agarwal, R. (2023). HTTP, WebSocket, and SignalR: A comparison of Real-Time Online Communication Protocols. In Lecture notes in computer science (pp. 128–135). https://doi.org/10.1007/978-3-031-44084-7_13

[37] Real-Time object recognition based on NAO humanoid robot. (2018, July 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/8726687

[38] Banaeian, H., & Gilanlioglu, I. (2021). Influence of the NAO robot as a teaching assistant on university students' vocabulary learning and attitudes. Australasian Journal of Educational Technology, 71–87. https://doi.org/10.14742/ajet.6130