An Augmented Reality Based Network Tic Tac Toe Game

Design Project report submitted in partial fulfilment of the requirements for the degree of B.Tech.

by

Adhesh Reghu Kumar
(Roll No: COE18B001)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KANCHEEPURAM

December 2021

Certificate

I, **Adhesh Reghu Kumar**, with Roll No: **COE18B001** hereby declare that the material presented in the Design Project Report titled **An Augmented Reality Based Network Tic-Tac-Toe Game** represents original work carried out by me in the **Department of Computer Science and Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram** during the year **2021**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Report is carried out under my supervision, and is worthy of consideration for the requirements of design project during the period November 2021 to December 2021.

Advisor's Name: Advisor's Signature

Abstract

In our design project, we develop an Augmented Reality Based Tic Tac Toe Application that can be played between users on a Network. The goal of our project is to design an application that enhances user experience of board games by integrating Augmented Reality and Network Communication. Our application enables distant users on the network to have in-person-like experience of board games by introducing virtual elements into the physical board. We have used digital image processing techniques as the building blocks towards the realization of this interactive augmented reality based game. The game allows users to connect to it through TCP/IP based network communication protocols. We have also integrated Artificial Intelligence into the game to allow single players to play against a computer AI.

Our application successfully combines the computer science concepts of Digital Image Processing, Augmented Reality, Artificial Intelligence, Network Communication and Software Development.

Acknowledgements

Firstly, I extend my gratitude to **IIITDM Kancheepuram** for providing with the opportunity to work on this project. I would also like to thank my guide, **Dr. Masilamani**, for his constant support and encouragement.

I also extend my gratitude to all who have helped me directly or indirectly in my work.

Contents

List of Figures

Abbreviations

AR	**A**ugmented **R**eality
VR	**V**irtual **R**eality
AI	**A**rtificial **R**eality
DIP	**D**igital **I**mage **P**rocessing
CN	**C**omputer **N**etworking
TCP	**T**ransmission **C**ontrol **P**rotocol
UDP	**U**ser **D**atagram **P**rotocol

For/Dedicated to/To my...

Chapter 1

Introduction

Augmented reality has been one of the biggest growing trends of the current decade with major advances being made in the field every year. Augmented Reality (AR) enhances user experience by augmenting computer generated virtual elements onto a physical world. A popular example of an AR application is the Pokemon Go game which was released in 2016 where users had to find and capture virtual pokemons that would appear in real world locations. AR, apart from gaming, also has found applications in various domains such as education, medicine, entertainment, navigation etc. Currently, the field of AR also experiences a massive push for advancement with Facebook's announcement of their 'Metaverse' vision, where Augmented Reality (AR) and Virtual Reality (VR) take the center stage as mediums of user experience.

The field of Computer Networking (CN), or networking in general, could be easily termed as one of the pillars of the connected modern civilization. Computer Network refers to a collection of computers that share resources or information. The best example of a computer network is the Internet. The applications of computer networking are so prevalent that it is difficult to imagine modern computer experience without them. Streaming, surfing, chatting, making video calls etc. are all examples of computer network applications.

Integrating concepts from the above two fields of AR and CN along with those from Artificial Intelligence (AI) and Software Development, in this design project, we develop an AR based Tic-Tac-Toe Network Game. The game developed for PC aims at providing players in-person-like experience while playing board games over the network. In this project, we have relied on Digital Image Processing (DIP) techniques, implemented using OpenCV, for the realization of the AR features in the game. Networking between players

has been implemented using Socket Programming, allowing players to connect with each other following the TCP protocol. In addition, we have also integrated AI into the application which enables single players to play against a computer AI. Implementation of the AI for the Tic-Tac-Toe is done using the Minimax Algorithm.

Overall, our application allows two users to connect using TCP/IP connection protocols and play a game of Tic Tac Toe in real time. The opponent's moves are augmented onto the player's grid using DIP techniques, implemented using OpenCV. A player can also choose to play with an inbuilt AI. The application is built for PCs, using PyQt, since the focus of the project was in development of the AR inspired concept and not on the mobile app development part.

Our project thereby successfully combines concepts from various domains of Computer Science such as Digital Image Processing, Augmented Reality, Artificial Intelligence, Network Communication and Software Development, including multi-threading.

1.1 Background

In this section we will give a brief overview of some of the concepts and principles used in our application. Let us begin our discussion with Canny Edge Detection, a mathematical algorithm used to detect edges in a given image.

1.1.1 Canny Edge Detection

Canny edge detection [1], as the name suggests, is an edge detection algorithm. Edge, in the context of image processing, is defined as the boundary between different segments of an image where there is significant change in the intensity of the pixels. The canny edge detection algorithm is a 5 step process:

- Step - 1: Apply Gaussian Filter for noise reduction and to smoothen the image
- Step - 2: Find the Intensity Gradients using Edge Filters (Sobel or Prewitt or Roberts)
- Step - 3: Non Maximum Suppression to thin out edges
- Step - 4: Double Thresholding to identify strong and weak edges

- Step - 5: Edge Tracking by hysteresis to transform certain weak edges into strong ones based on spatial conditions

Note: The input image to the algorithm must be in *grayscale*.

1.1.2 Hough Line Transform

Hough Transform [2] is a feature extraction technique used in DIP. Hough transform is very effective in detecting straight lines. For Hough transform, the straight lines are first represented in their normal form as:

$$r = x\cos\theta + y\sin\theta \tag{1.1}$$

where r is the length of the normal and θ is the angle between the normal and the x axis. The (r, θ) plane is also referred to as the *Hough Plane*. For every point on the plane, all straight lines that pass through it correspond to a sinusoidal curve in the plane, which is unique to that point. Therefore, if any two points form a line, their sinusoidal curve intersect. Hough transform thus detects the lines by finding the pair of edge points that have sinusoidal curve intersectsions larger than a given threshold.

Note: The input image to this algorithm is typically a Canny Edge processed image.

1.1.3 Networking Related Terminologies

Here we discuss some of the related network terminologies.

1. **Socket**: A socket acts as an endpoint for two way communication between processes running on a network. It is usually employed in client-server applications. A socket consists of an IP address (corresponding to the system) and a port number (corresponding to the process running the socket).
2. **Server**: The server is a program that passively listens to requests, processes the specified work and returns the results to the client.
3. **Client**: A client is a program that initiates a service request and connects to the server.

4. **IP Address**: IP (Internet Protocol) Address is the set of numbers that help uniquely identify a system in a network.

5. **Port**: Port is a number that helps uniquely identify a process within the system.

6. **TCP**: Transmission Control Protocol (TCP) is a connection oriented transport layer protocol. Being a connection oriented protocol, it establishes secure connection prior to any communication. TCP is a reliable protocol as it provides flow and error control mechanisms.

7. **UDP**: UDP is a transport layer protocol which is connection-less. It is typically used in for low-latency communication.

1.2 Motivation

Since the outbreak of the pandemic, social gathering with friends and family has decreased. Even though people connect through messages and video calls, the feeling of physical togetherness is absent. The traditional user experience models are unable to bridge this "invisible" gap. Augmented Reality can aptly address some of the shortcomings of the existing models by creating an interactive environment where users can have lived-in experience. An interesting application of AR is in gaming, especially board games. Board games are best enjoyed physically with a group of people. Augmented reality can provide an alternate to physical gathering, by virtual augmentation of players, while retaining the experience of physical interactions.

The goal of our project is in developing a concept application of such an AR based application for the tic-tac-toe game. We intend to understand the challenges and limitations that exist in development of such applications.

1.3 Objectives of the work

The objectives of the work done are as follows:

- To design and develop an AR based application for the Tic-Tac-Toe game.
- To implement network communication using *socket programming* to enable network users to play against each other.

- To build a *minimax* based AI feature in the application.
- To test the application under varied environments.
- To understand the challenges and limitations in the development of such applications.

Chapter 2

Methodology

Our project can be broken down into three main components:

1. Grid Detection
2. Networking
3. Tic-Tac-Toe AI

In this chapter, we will discuss in detail the working and theory behind each of these component.

2.1 Grid Detection

Detecting the tic-tac-toe grid from an image is the first important step in our application. A typical tic-tac-toe grid consists of two parallel horizontal and two parallel vertical lines intersecting each other. A sample tic-tac-toe grid is given in Fig 2.1. Because the grids are composed of straight lines, it is possible to detect the contours of the grid using an inbuilt cv2 function, *cv2.HoughLinesP()*, followed by appropriate post processing.

The objective of our grid detection algorithm is to find the vertices of the horizontal and vertical lines, followed by their points of intersections to finally get the coordinates of each of the 9 relevant sub grids from the input image.

Our grid detection algorithm consists of the following steps:

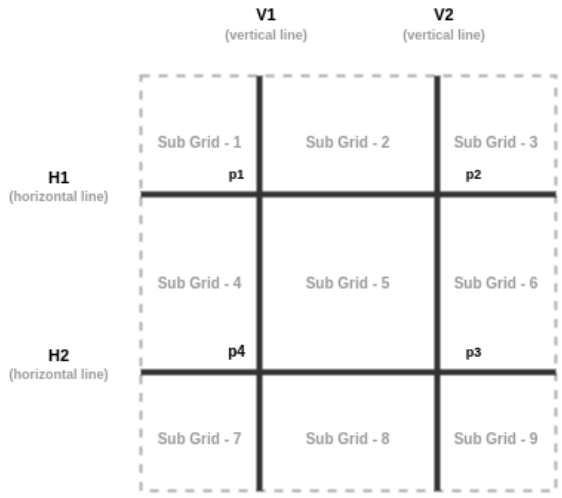

FIGURE 2.1: A Sample Tic-Tac-Toe Grid

1. Pre-process the raw image:
   - Resize image appropriately
   - Convert RGB image into Grayscale
   - Apply Gaussian Blur to reduce noise
   - Apply CannyEdge function to obtain the edges in the image
2. Run Hough Transform on the edge detected image to get vertices of the lines detected
3. Find the horizontal and vertical lines as follows:
   - **H1** = line having minimum x coordinate in its starting coordinates
   - **H2** = line having second minimum x coordinate and is at a $distance > threshold$ from H1
   - **V1** = line having minimum y coordinate in its starting coordinates
   - **V2** = line having second minimum y coordinate and is at a $distance > threshold$ from V1
4. Sort H1 & H2 according to their *y-coordinates*, and V1 & V2 according to their *x-coordinates*
5. Find the points of intersection of H1-V1 as ***p1***, H1-V2 as ***p2***, H2-V1 as ***p3*** and H2-V2 as ***p4***.

The vertices of lines H1, H2, V1 and V1, along with points of intersections p1, p2, p3 and p4 can be used to find the relevant coordinates of each of the 9 sub-grids.

2.2 Networking

The network game in our project is implemented as a client-server application using socket programming. A **client** is a program that initiates a service request and connects to the server. The **server** is a program that passively listens to requests, processes the specified work and returns the results to the client. The communication between the client and server follow well-defined communication guidelines called **protocols**. Usually the communication using sockets follows either TCP or UDP protocol.

TCP (Transmission Control Process) protocol is a connection oriented protocol which is a reliable protocol that ensures that the packets sent are not lost and arrive in an orderly fashion. They follow error and flow control mechanisms. **UDP (User Datagram Protocol)**, on the other hand, sends independent packets of data and provide no guarantee that packets won't be lost in transmission. They also lack proper error control mechanisms. Nevertheless UDP is useful for sending short packets fast. We follow TCP based socket connections for our network game. Fig 2.2 shows the sequence of socket API calls and data flow for TCP.

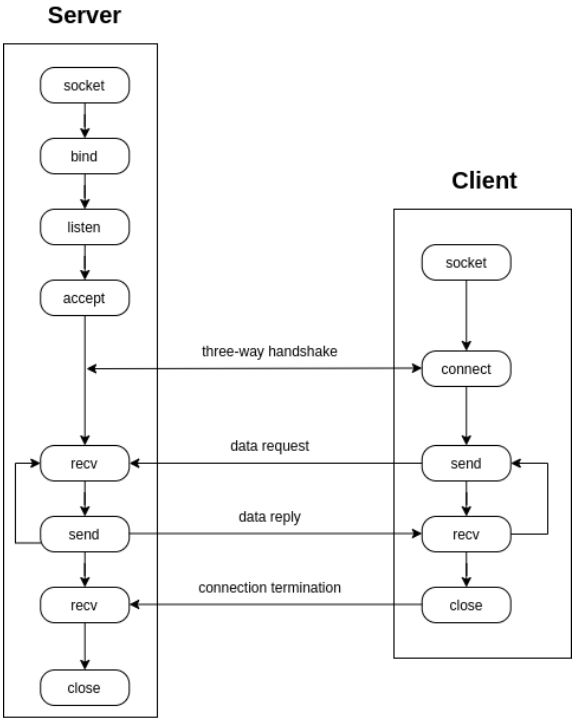

FIGURE 2.2: Socket calls and data flow for TCP

In our project, the clients are the individual instances of the GUI based application. The client program connects to the *gameroom (server)* when the user enters the appropriate connection parameters on the GUI interface given.

The client then program performs the following network communications:

- On making any move in the game, **sends** a message to server which is then forwarded to the opponent.
- Continually **listens** to any incoming messages from the server on a separate thread

Whereas the server program performs the following network communications:

- Passively wait for 2 players to join the server. Once 2 clients have joined, it initiates a new game session.
- On the start of a game session, it **broadcasts** parameters related to the game to each client.
- It then continually **listens** to any incoming message from any of the client program and **forwards** it to the other client.

The data flow between the clients and the server is shown in Fig 2.3.

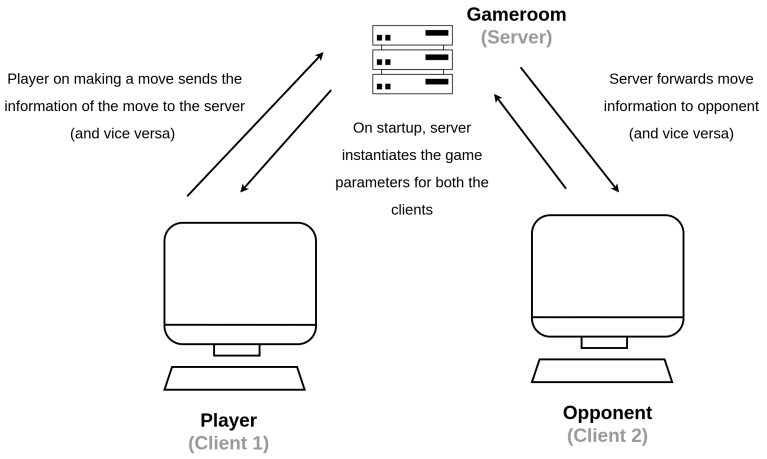

FIGURE 2.3: Data flow between server and client programs

2.3 Tic-Tac-Toe AI

The AI for the Tic-Tac-Toe game is implemented using **Minimax** [3] algorithm, an adversarial search algorithm which generates and explores game trees to find the optimal move for a player. Tic-Tac-Toe is a simple 2-player game played on a 3×3 grid and has only **255,168** possible game configurations. Therefore, the outcome given a state can be determined by just exploring future game configurations in the game tree.

Implementing minimax for Tic-Tac-Toe is straightforward. For each board state, we will recursively generate the game tree by exploring all possible moves until reaching terminal state. Upon reaching terminal state, we evaluate the score as follows: **10** if player wins, **-10** if opponent wins and **0** if the game is tied. Based on the score, the player makes his/her optimal move. The pseudo-code for the minimax algorithm for Tic-Tac-Toe is as follows:

Algorithm 1 Minimax algorithm for Tic-Tac-Toe Game

```
procedure MINIMAX(board, depth, isMaximiser)
    if board status is terminal then
        return score
    end if

    if isMaximizer then
        for each board state do
            score = minimax(board, depth+1, False)        ▷ compute score recursively
            bestScore = max(score,bestScore)
        end for
        return bestScore
    else
        for each board state do
            score = minimax(board, depth+1, True)         ▷ compute score recursively
            bestScore = min(score,bestScore)
        end for
        return bestScore
    end if
end procedure
```

Chapter 3

Work Done

In this chapter, we will discuss in detail how the components discussed in previous chapter is put together to build a cohesive application. We begin with the description of the GUI component of the application, and then explain the internal flow of the application.

3.1 GUI Component

The GUI component of our project is made using PyQt [4] python package for desktop application. The application consists of a single window which is simple and pretty intuitive to use. The GUI of the application can be visually divided into 4 sections (shown in Fig. 3.1):

1. **Header**: It consists the title and a short description of the application.
2. **Controls**: Allows user to enter network and camera related parameters and control the application through the several control buttons.
3. **Camera Display**: Display area for the real-time camera feed. Occupies maximum space in the application.
4. **Status Bar**: Displays the current status of the application.

A dark color scheme of black background with white text has been opted for the GUI. This helps provide a contrast against the white color of the paper displayed dominantly on the *Camera Display* section, thereby bringing a balance in the design.

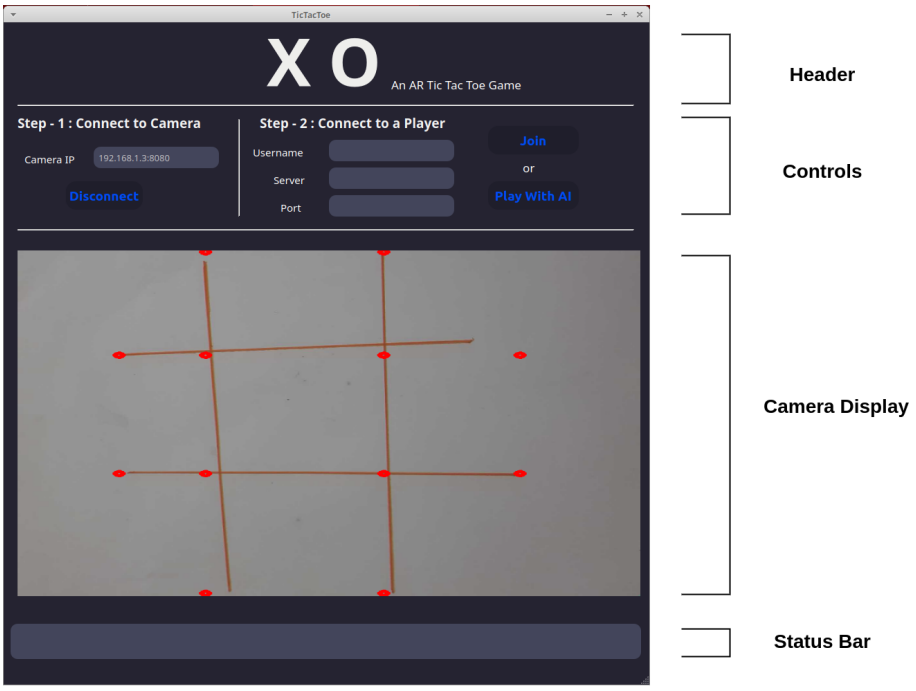

FIGURE 3.1: A look at the application GUI

Let us now discuss a bit more regarding the inputs and buttons available in the Controls Section of the GUI. The *Controls Section* can be further visually divided into two sub sections:

1. Connect to Camera:
    - **Camera IP (Input)**: expects IP address of the camera stream. If the inbuilt webcam is being used, the user can just enter '0'.
    - **Connect/Disconnect (Button)**: On clicking the connect button, the video stream is displayed on the *Camera Display* section and the button toggles to Disconnect functionality, which upon clicking ends the video stream display.
2. Connect to Player:
    - **Username, Server, Port (Inputs)**: expects username, the IP address and port number of the server respectively *(Only Applicable if playing a Network Game)*
    - **Join (Button)**: validates user inputs and connects to the server.
    - **Play With AI (Button)**: connects the player with the inbuilt AI

3.2 Program Flow

In this section, we explain how the functional components discussed earlier are put together to form a working application. But before we look at the complete program flow, let us quickly introduce some more minor components of the application.

3.2.1 Player Movement Detection

Once a player has made his/her move, the program must be able to identify the movement from the image of the board. This can be done by extracting the 9 sub-grids from the image by passing it through the grid detection algorithm described earlier. The 9 sub-images are then passed through template matcher and circle detection algorithms to obtain the new board configuration. It is then compared with the previous board configuration to check for illegal moves and then update of board happens.

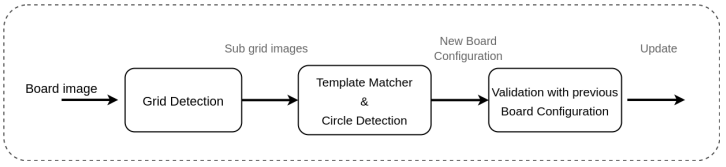

FIGURE 3.2: Movement Detection

3.2.2 Augmentation

The final step in the game loop is the augmentation of the opponent's move onto the player's grid. To do so, we require the coordinates of the grid in which the opponent's move was made. This coordinate information can be obtained from the grid detection algorithm. With this coordinate information, we can easily overlay the appropriate symbol ('X' or 'O') onto the board image using OpenCV functions.

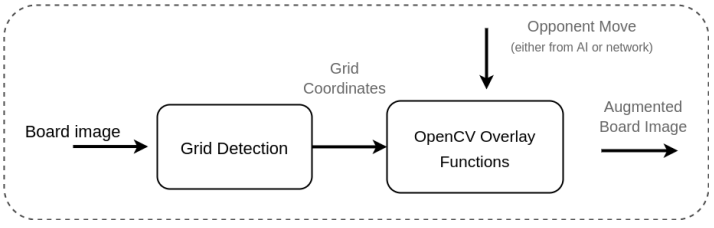

FIGURE 3.3: Augmentation Procedure

3.2.3 Program Flow

In this section, we present entire program flow of our application.

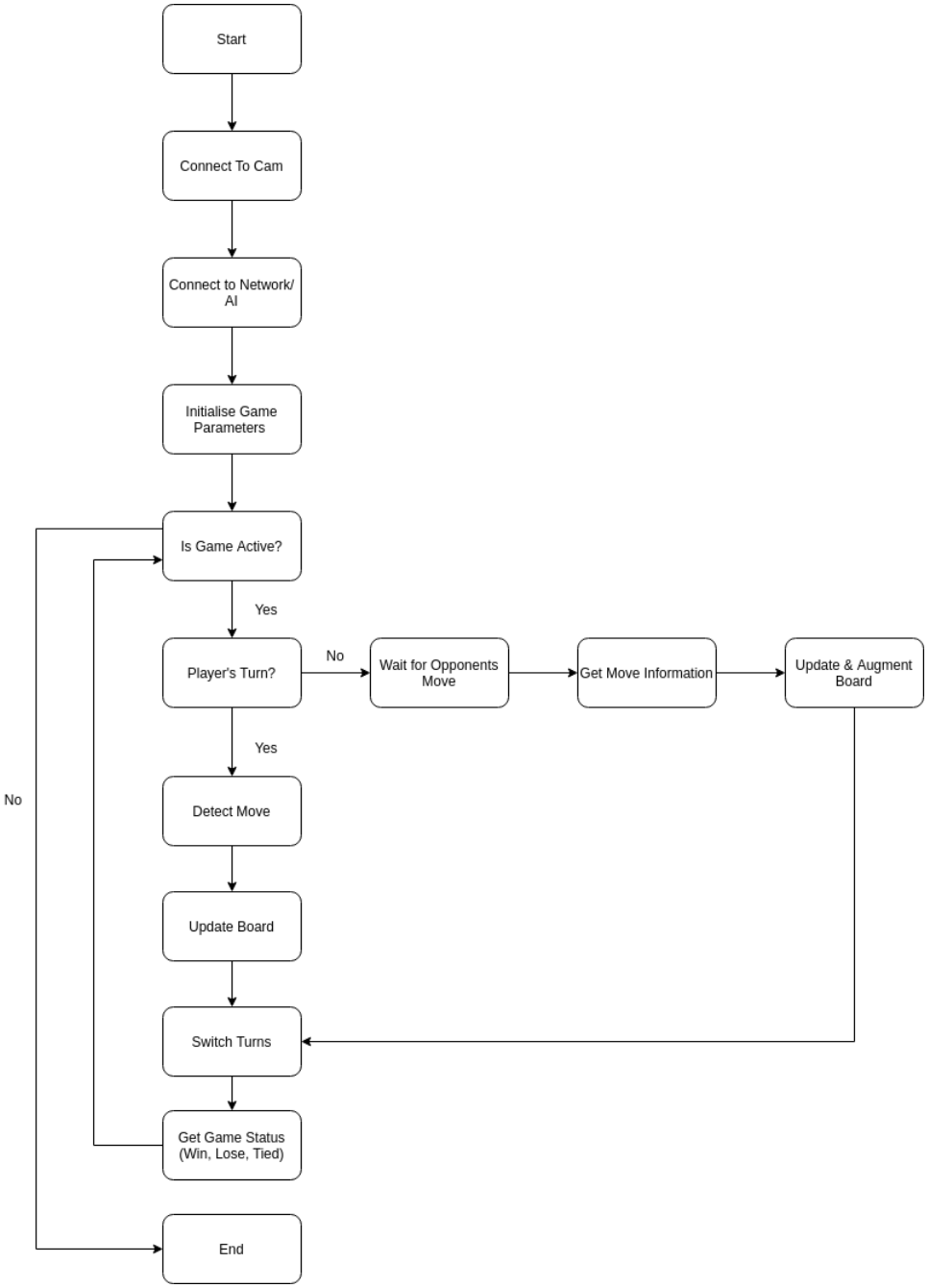

FIGURE 3.4: Program Flow

Chapter 4

Gameplay and Discussions

In this chapter, we demonstrate the gameplay of our Tic-Tac-Toe game and discuss the challenges faced in the development of the application. We also discuss the limitations of the current application.

4.1 Gameplay

Our application consists of two player modes:

1. **Against Network Opponent**: Where the user can connect to a *gameroom* server and play with another player in the network.
2. **Against AI**: Where user can play against an inbuilt Tic-Tac-Toe AI. This mode can only end in either a '*Tie*' or '*Lose*' condition.

Both play modes follow the program follow described in the previous chapter. The only difference is in the Opponent's decision making process. In the network game, the player program continually listens to messages from the opponent (forwarded through the server) on a separate thread. As soon as it receives the move information from the opponent, it updates it's board. On the other hand, decision making when playing against the AI is instantaneous. The AI, using minimax algorithm, computes the optimal move and updates the player board accordingly. Now, let us take a glimpse at some the gameplay pictures of our application.

4.1.1 Demo

Fig. 4.1 shows the game flow of our application.

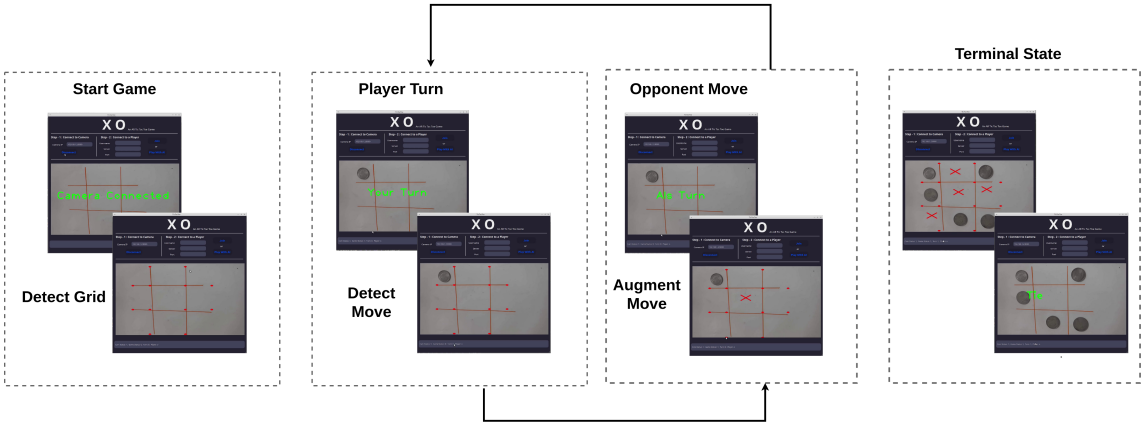

FIGURE 4.1: Game Flow (Against AI)

Detailed demonstration videos of our game can be found in the below links:

1. **Against AI**:
   `https://drive.google.com/file/d/1J1smCIjh3VSZi-GalTGYRYFH4CC1t1Sk/view?usp=sharing`
2. **Against Network**:
   `https://drive.google.com/file/d/1Pc_VoFBLrrSuC2EvED7oQ8d5Wf1VBHKe/view?usp=sharing`

4.2 Challenges

Our application deals with real-time video processing, augmented reality and computer networking, all of which present different challenges on implementation. Some of the notable challenges that we faced are:

- Achieving low latency processing of the video was a difficult task, often causing the application to crash. A solution to this is to set an appropriate time-delay (frame rate) between the consecutive frame captures.
- Stability of the camera is a very important factor for accurate grid detection. If the frame captured by the camera changes frequently, it becomes difficult to process

the image and detect the grid. We can resolve this issue by keeping a copy of the previously detected grid points and comparing them with the new grid points. If they compare well within a given threshold, we can use the newly obtained grid points. This takes into account the principle of temporal locality.

- Management of the multiple threads in the program can be challenging and has to be done with care.

4.3 Limitations

Even though our application is able to fairly fulfill all of our objectives, there are still a few limitations in the current version. Some of the limitations are as follows:

- Grid detection is done using conventional OpenCV methods. Even though these methods are very fast, they can still fail in certain conditions. There are also cases where false lines and points are detected in a noisy image.
- Several threshold parameters are still manually input to the algorithms. Ways to automate this must be looked into.
- We currently use circle detection and template matching algorithms to detect opponent move. Even though this is fast and effective for most cases, it still does not consider the variability in input. Circles need not be perfect and crosses can be drawn in a variety of ways. Hence we have to look into alternate ways to detect these symbols from an image.

Chapter 5

Conclusions and Extensions

5.1 Conclusions

In this project, we successfully implemented an interactive Augmented Reality based Tic-Tac-Toe Network Game. Through our project, we show how AR can be used to develop applications that enhance user experience, especially in the domain of gaming. Our report details out the methods and algorithms used in the development of the application. Our project was able to successfully integrate concepts from the various domains of computer science such as *augmented reality*, *digital image processing*, *computer networking*, *artificial intelligence* and *software development* into a cohesive application. From the project, we also gained valuable insights regarding the limitations and challenges that can be expected when developing an AR application, or an application in general.

5.2 Future Work

In future, we plan to port the current PC based application onto a mobile platform. We also intend to overcome the current limitations of our application by making improvements on the algorithmic front. We would also like to further experiment developing AR based applications for other similar pen-and-paper games such as '*Dots and Boxes*', '*Battle ship*', '*Hangman*'etc.

Bibliography

[1] "Canny edge detection." [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

[2] "Hough line transform." [Online]. Available: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

[3] "Minimax," Oct 2021. [Online]. Available: https://en.wikipedia.org/wiki/Minimax

[4] "What is pyqt?" [Online]. Available: https://riverbankcomputing.com/software/pyqt/intro