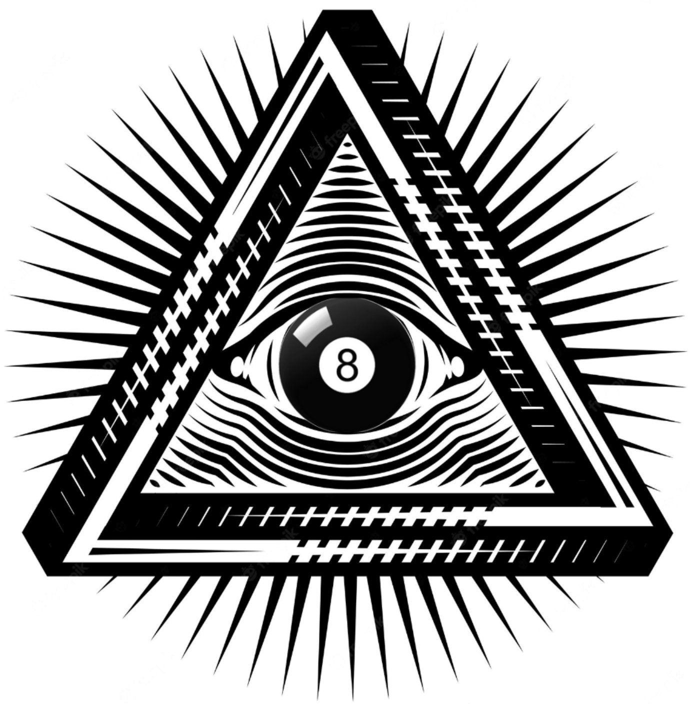
VISION

Visually Impaired Spatially Interactive Orientation Network



Group 14 Authors:

Aaron Crawford
Computer Engineering

Alexander Parady
Electrical Engineering

Arsene Landry Tatke
Electrical Engineering

Noah Harney
Computer Engineering

Mentor:

Dr. Chung Yong Chan

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY **1**

2. PROJECT DESCRIPTION **2**

2.1 Project Background and Goals 2
2.2 Project Motivation 3
2.3 Project Function 4
2.4 Project Objectives 5
2.5 Required Specifications 7

3. RESEARCH **10**

3.1 Similar Projects 10
3.2 Relevant Technologies 10
3.2.1 Billiards Artificial Intelligence 10
3.2.1.1 Simulation Tools 10
3.2.1.2 Simulation Tool Modifications 12
3.2.1.3 Different Implementations of Shot Selection Algorithms 12
3.2.1.4 Computation of Shot Selection Algorithm 18
3.2.2 Computer Vision 19
3.2.2.1 Computer Vision Software Options 19
3.2.2.2 Computer Vision Preprocessing 21
3.2.2.3 Computer Vision Algorithms 25
3.2.3 Visual Impairment Assistive Technology 31
3.2.4 User Localization 33
3.2.4.1 RFID And Bluetooth 34
3.2.4.2 Sensors 41
3.2.4.3 Localization Algorithms 46
3.2.5 User Guidance 47
3.2.5.1 Audio Outputs 48
3.2.5.2 Physical Sensory Outputs 50
3.2.5.3 Guidance Algorithms 52
3.2.6 Feedback System 53
3.2.7 Direct User Commands 56
3.2.7.1 Control Interfaces 57
3.2.7.2 Audio Commands 58
3.2.8 Visual Display 58
3.2.9 Absolute Orientation 61
3.2.9.1 Cue Displacement 61

3.2.10 Test Cases 61
3.2.10.1 Game Modes 61
3.2.10.2 Shots Supported by VISION 63
3.2.10.3 Physical Limitations 64
3.2.11 Processing Unit 65
3.2.12 Communication Methods 70

4. RELATED STANDARDS & DESIGN CONSTRAINTS 72
4.1 Related Standards 72
4.1.1 Wired Communication Standards 72
4.1.2 Wireless Communication Standards 73
4.1.3 Connection Standards 74
4.1.4 Programming Standards 74
4.2 Design Constraints 75
4.2.1 Economic Constraints 75
4.2.2 Environmental Constraints 76
4.2.3 Social and Political Constraints 77
4.2.4 Ethical Constraints 77
4.2.5 Health and Safety Constraints 78
4.2.6 Manufacturability Constraints 78
4.2.7 Sustainability Constraints 78

5. SYSTEM HARDWARE DESIGN 80
5.1 Billiard Table 80
5.2 Camera 81
5.2.1 Computer Vision Camera 81
5.2.2 Computer Vision Camera Mounting 84
5.3 Visual Display 85
5.4 Localization System 85
5.5 User Guidance System 88
5.5.1 Audio Array Design 89
5.5.2 Positioning Method 91
5.5.3 Orientation Method 91
5.6 User Control Interface 92
5.7 Communication Network 92
5.7.1 Communicating Systems 93
5.7.2 Communication Protocols 94
5.8 Processor Selection 95

6. SYSTEM SOFTWARE DESIGN **98**
6.1 Pool AI 98
6.2 Computer Vision System Software Design 103
6.3 Navigation Algorithm Design 109
6.3.1 Localization Algorithm Design 109
6.3.2 Guidance System 111
6.4 Web Interface 111
6.4.1 Output Image Generation 116
7. SYSTEM FABRICATION **117**
7.1 PCB Design 117
7.1.1 PCB Design Philosophy 117
8. SYSTEM TESTING PLAN **119**
8.1 Hardware Testing 119
8.1.1 Guidance Testing 119
8.1.2 BLE Testing 120
8.2 Software Testing 121
8.2.1 Shot Selection Algorithm Testing 121
8.2.2 Computer Vision Software Testing 122
8.2.3 Feedback System Software Testing 124
8.2.4 Localization Software Testing 125
8.3 User Testing 126
9. ADMINISTRATIVE CONTENT **128**
9.1 Project Budget 128
9.1.1 Bill of Materials 128
9.1.2 Project Financing 129
9.2 Milestones 129
10. PROJECT SUMMARY & CONCLUSION **133**
Appendix A: Copyright Permissions **134**
Appendix B: Code Segments **145**
Appendix C: References **146**

LIST OF FIGURES

Figure 2.1 Project Block Diagram	5
Figure 2.2 House of Quality Analysis	9
Figure 3.1 Shot Planner Diagram	13
Figure 3.2 Neural Network Work for State Set with Three Balls	16
Figure 3.3 Neural Network for State Set with 3 Balls (Broken Into Two Networks)	17
Figure 3.4 Ideal Distribution of Thresholding on Image	23
Figure 3.5 Image Thresholding to Isolate Region of Interest	24
Figure 3.6 Canny Edge Detection on an Image	26
Figure 3.7 Detection of Overlapping Circles	29
Figure 3.8 Epsilon Value on Algorithm Output	30
Figure 3.9 Previous System Indoor Localization Design	32
Figure 3.10 AD-172u7 UHF RFID Tag and Inlay	36
Figure 3.11 Simplified Model of Trilateration	39
Figure 3.12 Bluetooth BLE iBeacon (BC021-MultiBeacon) with Motion Sensor	41
Figure 3.13 Model and Dimensions of Compact Housing HRXL-MaxSonar Model	43
Figure 3.14 VL53L0X Time-of-Flight Ranging and Gesture Detection Sensor	44
Figure 3.15 Localization Algorithm Array Scheme	47
Figure 3.16 Audio Based Navigation Mechanisms	49
Figure 3.17 Maptic Haptic Feedback Apparatus	51
Figure 3.18 HandSight Haptic Feedback Apparatus	51
Figure 3.19 Maze Traversal Example	52
Figure 3.20 Force Resistive Sensor	54
Figure 3.21 RFID Tag Embedded in Golf Ball	55
Figure 3.22 Feedback System Shot Results	56
Figure 3.23 TV Remote for the Visually Impaired	57
Figure 3.24 8-Ball Features Supported By VISION	62
Figure 3.25 Cue Contact Point	64
Figure 5.1 Blue Wave's Fairmount Table (Top) & Rack's Crux 55 Table (Bottom)	81
Figure 5.2 Example of Fixture Mounted Camera	85
Figure 5.3 Pool Table with BLE Beacons	86
Figure 5.4 Prototype Speaker Activation Design	90
Figure 5.5 Designed Speaker Array	90
Figure 5.6 Bidirectional Guidance Possibilities	91
Figure 5.7 Worst Case Margin of Error Estimation	92
Figure 5.8 Tentative Communication Network	94
Figure 5.9 Jetson Nano Device Connections	97
Figure 6.1 Example of Reachable Shot Issue	99
Figure 6.2 Shot Angle Projection	100
Figure 6.3 Shot Angle Quadrant	100
Figure 6.4 High-Level Overview of Shot Selection System	103
Figure 6.5 High-Level Overview of Computer Vision System	104
Figure 6.6 Computer Vision Implementation	108
Figure 6.7 Senior Design Dashboard File/Folder Structure	115

Figure 6.8 Dashboard Layout 116
Figure 7.1 PCB Design Block Diagram 119

LIST OF TABLES

Table 2.1 Requirement Specifications 8
Table 3.1 Comparison of RFID Technologies 34
Table 3.2 Comparison of Different Ultrasonic Sensors 42
Table 3.3 Summary of Processor Offerings 67
Table 3.4 Performance Results of Benchmark Testing 69
Table 5.1 Summary of Camera Options 83
Table 5.2 Summary of Power Consumption in ESP32 87
Table 5.3 Comparison of Communication Interfaces 94
Table 6.1 Comparison between R and Python 113
Table 9.1 Bill of Materials 129
Table 9.2 Senior Design 1 Project Documentation Milestones 131
Table 9.3 Senior Design 1 Project Design Milestones 132
Table 9.4 Senior Design 2 Project Design Milestones 133

1. EXECUTIVE SUMMARY

Every day tens of thousands of people around the world struggling with disabilities have difficulty enjoying aspects of life that many people take for granted. People that yearn to walk on, touch, smell, and see the world around them in ways that they cannot. In more recent years, technology has certainly expanded the freedom of impaired individuals, but there is still a significant amount of work to be done. VISION allows people struggling with visual impairments to be able to play a game of 8-ball billiards without the need for additional human interaction. The goal is to allow the visually impaired to participate in a common pastime while also feeling a sense of independence.

The idea for VISION began as an idea for making an autonomous billiards training agent that a billiards player could utilize to improve their performance. Although this was an innovative idea that can certainly help billiards players, the idea lacked a true societal impact. After much thought, the idea arose to implement a system that performed all of the tasks a visually impaired player would not be able to perform. VISION is quite literally the vision of a player that locates, localizes, and strategizes the game for a user.

VISION incorporates some of the most modern technology to implement a system that is robust yet simple enough for people without an extensive background in electronics to utilize. Upon starting the system, VISION uses a camera to capture the current state of the billiards table. Computer vision algorithms then identify all of the billiard balls on the table and determine the position and color of the balls. An artificial intelligence algorithm is then used with the billiard ball locations to determine the best shot a user can take. VISION will then track the location of the user and provide audio instructions to the user to guide the player to the correct position for the shot. Once in the correct location, the user will be guided to face in the appropriate direction to take a shot.

At this point, VISION will send information regarding the ideal shot and user positioning to a related project named SCRATCH to complete the actual shot. SCRATCH is a project working in conjunction with VISION that is responsible for the fine-tuning of a user shot. Once a player has made a shot, VISION will then be able to determine the outcome of the shot and audibly notify the user of the results.

VISION is a large, complex project that incorporates many relevant topics in computer science and electrical engineering to create a product that has never been made before. VISION is certainly an ambitious project, but the team members are committed to widening the inclusivity of one of America’s favorite pastimes.

2. PROJECT DESCRIPTION

2.1 Project Background and Goals

Billiards is a collection of many different games played with a billiards table, cue stick, and several colored billiard balls. The objective of a billiards game varies depending upon what specific game is played, but the typical goal is to use a cue stick to pocket a targeted game ball. Every specific billiard game introduces rules and requirements that make sinking a shot more difficult than it may seem. One of the more common billiard games, and the focus of this project, is 8-ball pool. The goal of VISION is to design and implement a system that allows individuals suffering from visual impairments to become capable of playing a game of 8-ball billiards.

Billiards was selected as the game of choice because of its significant complexity compared to other games such as chess. Chess is a game commonly associated with masterful planning that requires crafting moves multiple turns in advance to be successful. Although chess certainly is a complex logic game, it is a discrete problem in terms of computation. Chess has a fixed number of locations on the board, a specific number of pieces with strict rules about where they can move, and a finite number of possible ways for the game to progress. All of these reasons have led chess to become a commonly studied problem in computer science. There are many computer programs and algorithms for chess that are quite good at the game. There has been much less research conducted on creating a robust billiards program. Furthermore, there does not appear to be any billiards-style game developed specifically for the visually impaired.

Like chess, billiards also requires players to plan their moves many turns in advance in an offensive or defensive manner. An offensive move is when a player tries to sink as many balls as possible while a defensive move is when a player tries to put their opponent in a position such that their opponent cannot complete a shot. The careful shot selection necessary for billiards is significantly more involved than the equivalent chess decision because there is an infinite number of positions that the state of the billiards table can be in. The billiard balls can arrange themselves in any position on the table at any point during the game, the same cannot be said for chess. There are many ways for a game of billiards to progress, and it can oftentimes be difficult to know what the best shot to take is given the current state of the game.

For the vast number of chess programs and significantly fewer billiards programs that have been developed, nearly all of these projects have been software implementations of the game. The programs that were created were designed to be used for virtual games, not physical chess boards or actual billiards tables. The versions of billiards games prove that a software system can be used to implement a game of pool. One of the goals of VISION is to expand upon previous work by using an actual game of billiards, rather than a simulation of the game.

The success of VISION will be determined if an individual dealing with visual impairments is able to successfully compete in a modified game of billiards. With the help of VISION, a user should have the billiards table represented algorithmically and have the best shot determined for them. The user's location should be tracked and used to navigate the user around the billiard table. The result of the user's shot should then be displayed in a program to spectators around the room. If all of these individual goals are met, VISION will be a success. VISION should be compact and portable so that the system can be disassembled, moved, and assembled in a timely manner.

2.2 Project Motivation

The motivation of VISION is to develop a systematic way to represent a real-life game of 8-ball pool computationally and then develop an elegant way to guide a visually impaired user through the best shot for them to take to win the game. VISION is a tool that can leverage the power of modern technology to help improve the inclusiveness of one of society's most popular pastimes.

For VISION to truly have an impact, the team decided to develop it in a way that allows individuals dealing with visual impairments to develop a sense of autonomy. There are not many games that have support for people dealing with disabilities. It can be difficult for some individuals to feel included when they are not able to participate in the same pastimes as their friends and family. Globally, about 295 million people have a case of near or far distant visual impairment. In addition to this, about 43 million people worldwide suffer from complete blindness. One of the biggest troubles they face in their everyday life is having their freedom limited by moving in an obstructed or limited environment where spatial awareness is preventing them from being able to engage in their daily activities.

A lot of systems are in place in different media to help counteract or ease these issues to breach issues of orientation, localization, and way-finding through different technologies. Navigation technologies or electronic travel aids have been the backbone when it comes to developing technologies to help visually impaired people bridge the way for more specific applications such as the one we are working on for this project. Similar to the goal of our project, a lot of sports rules have been adapted and modified to develop games that are more inclusive to visually impaired individuals. For instance, beep baseball where the bases beep to let the players know which direction they need to go in, or soccer where the regular ball is replaced by an audible ball. We will use these and similar concepts as a motivation and a basis to determine which objectives and checkpoints are needed to make VISION an impactful visually impaired technology. Our team is motivated to broaden the inclusiveness of billiards by creating a system that leverages technology to plan, strategize, and see for a player.

2.3 Project Function

A visually impaired individual that is using the VISION system will be able to have the system locate all of the billiard balls and determine the optimal shot for them to win the game. VISION will actively track the user and be able to guide the user to the required location through audio instructions. The system will provide instructions to the user to ensure that they are positioned in the general direction of the cue ball. At this point, VISION's job is complete and the SCRATCH program (group #17) will take over. VISION will provide SCRATCH with the optimal shot angle, required force, and location of the cue ball.

There is certainly a concern when two projects are interrelated with each other in Senior Design. It would not be fair if one project's failure leads to the failure of the other project. With the help of our mentor, the teams designed their projects in a way that minimizes interaction between the two projects. VISION will transmit three quantities to SCRATCH and the three values can easily be artificially constructed if needed. The SCRATCH team does not need to transmit any information back to the VISION team. If the VISION team fails to complete their project, the SCRATCH team can craft inputs that the VISION team should have provided. If the SCRATCH team fails to complete their project, the VISION team will lay the groundwork for future work. VISION will detect billiard balls, find the optimal shot, track the user, guide the user to the appropriate position, and position the user in the appropriate direction.

The VISION team must design a system that is lightweight and able to be moved between different locations. The system must be designed so that it can quickly be disassembled and reassembled so the team can work on the project in a variety of locations and environments. The mobility of the system will also be helpful when demonstrating VISION to others and must be set up in different locations.

VISION is a large project that incorporates a large variety of technology into a single, user-friendly system. The central processor for the system will be a powerful, computer-like processor capable of running computer vision and artificial intelligence algorithms. There are many systems that must be integrated for VISION to work properly. Figure 2.1 below shows a block diagram of all of the systems needed.

All systems will be controlled by the powerful central processor shown in the middle of the diagram. The processor will ask the computer vision system to capture the current state of the board with a camera and transform the physical billiards game into data expressed in a computational way. The shot selection algorithm is then used to determine the best shot to take given the current state of the table. The information regarding the best shot to take will then be sent to the SCRATCH team and used internally. The shot information is used by the user localization and user guidance systems to determine where the user is and how to guide them to the proper location. Once the user is in position, the control will be transferred to the SCRATCH team to take the actual shot. Once the shot has been executed, VISION will take back control and determine the results of the player’s shot. The results are displayed on a monitor and announced through an audio system.

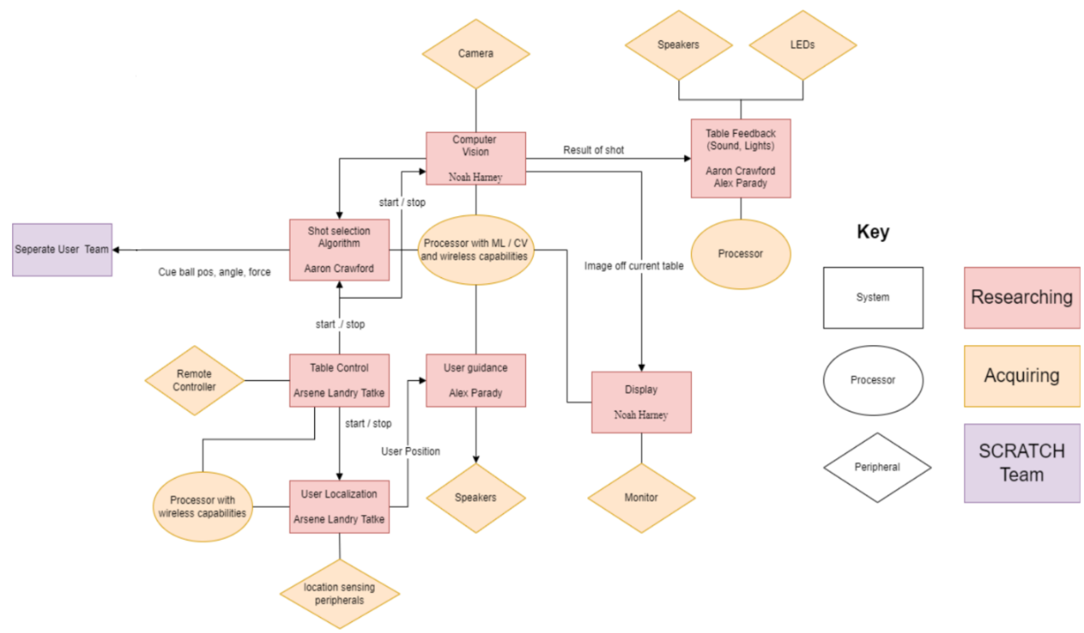


Figure 2.1 Project Block Diagram

2.4 Project Objectives

VISION should encompass a system that allows a visually impaired individual using our system to be detected around the pool table. Before a game begins, this will allow us to determine where the user is and bring them to the position where they would make their initial shot from the cue ball to the stack of balls. After every turn and during their subsequent turns, the same system should be once again able to detect the user to know where they are with respect to either the cue ball or within a set coordinate system determined by the project and with respect to the table. VISION might also encompass other localization schemes as either outlined during the research section or in the future considerations section. The first one would require a system that expands the range of localization from anywhere in the room containing the pool table. The second one would require a system that detects any obstacles around the user in the room containing the pool table or around the pool table itself. Both of these localization schemes are secondary to the main one requiring the user to be located around the pool table itself which we deem to be the most relevant objective when it comes to allowing the visually impaired user to pursue a game of pool.

VISION should encompass a system that captures the current state of the pool table at every point during the game, that is, at the start of a game, and every round during the game. This system would then process the image to isolate the pool table from any sort of background present in the image. The system should be able to detect, isolate and localize the billiard balls present on the pool table. The system should be able to differentiate the cue ball, the eight ball, the player balls, and the opponent balls. This system should also be able to render a mapped image of the billiard table layout with the balls at the right position on the rendered image to continuously reflect the current state of the game and will be used in other systems described below.

VISION should encompass a system that computes the optimal shot that the user, visually impaired or not, can make based on a shot selection algorithm. This will involve making some considerations and assumptions described in later sections of the document due to the multitude of factors coming into play such as the skill level of the user, outside interference during the shot, and other relevant factors. We will outline the optimal output that this algorithm will need to provide after considering different options such as how much force would need to be put to make the shot, the positioning of the user's hand on the cue stick, the angle from the base of the table to the cue stick, user posture, and other related metrics.

VISION should encompass a system that navigates the visually impaired user to the necessary position that the aforementioned algorithm would determine, the position in which he/she has the best odds to make a ball. This system will rely on the previous systems to determine what the optimal location of the user will be to take the desired shot. This calculation will be needed after every shot the user takes. Our system should also be able to navigate the visually impaired user through a non-visual mechanism such as audio, tactile inputs, or similar methods. This system should be able to outline clear commands or properly explain commands to provide concise instructions to the user.

VISION should encompass a system akin to a dashboard available for all users around the pool table, player, or spectators. The system should have the rendered image of the pool table mentioned above, as well as different statistics about the current game updated in real time. The system should hence have a way of tracking, storing, interpreting, and displaying information about the ongoing game. Considerations would need to be taken to determine the optimal way to display different information about the game in an eye-catching and intuitive manner for all users. For a visually impaired user, this system might include audio outputs to vocalize shot results and important information about the game progression. Considerations would need to be taken to avoid audio overload if audio is also being used as a way to navigate the user. This system would need to be presented on a medium or display readily available for everyone around the table without obstructing the game in session.

All of the components of VISION should be modular and able to be individually tested before being integrated with the entire system. The components of VISION should also be able to be assembled and disassembled quickly. The entire system should be able to be transported in a sedan so that there is no problem moving the system from one location to another.

VISION is currently a self-funded project and also would like to be made affordable enough for someone to reproduce themselves. For these reasons, the team would like to keep the project under $800, so each member will not have to contribute more than $200. If significant changes are needed, the budget may need to be reevaluated.

2.5 Required Specifications

The previous sections describe the goals, objectives, and motivation behind VISION. To transform VISION from an idea into an actual project, requirement specifications must be clearly defined. These requirements are what the VISION team believes are necessary to bring the project to life. These requirements serve as a contract between the team members and the senior design advisors clearly stating what the project will be able to do. The success of VISION will be based on meeting the requirements specified in table 2.1.

Requirement	Description
1.1	Locate up to 10 billiard balls on the billiards table
1.2	Differentiate between green, blue, black, and white billiard balls
1.3	Locate all balls in an (x,y) coordinate system within 15 pixels
1.4	Locate all six pockets in an (x,y) coordinate system within 15 pixels
1.5	Latency of the computer vision system does not exceed 5 seconds
2.1	Latency of the shot selection algorithm does not exceed 5 seconds
2.2	Shot selection algorithm will produce a shot suggestion with a minimum specificity of 5 degree increments
2.3	Shot selection algorithm will produce a shot suggestion with a minimum specificity of 3 force levels
3.1	Latency of the user localization does not exceed 10 seconds
3.2	Accuracy of the user localization is within 1 foot of true location
3.3	Localization aid should work independently of the surroundings
4.1	Position user within 1 foot of desired standing position for shot
4.2	Orient user within 15 degrees of desired shooting direction
5.1	VISION can be assembled or disassembled in less than 30 minutes
5.2	The total cost of VISION should not exceed $800
5.3	The product's audio aids will support the English language
5.4	Battery-powered devices used within the system should be viable for 1 year

Table 2.1 Requirement Specifications

To best quantify the correlation of various portions of VISION's defined deliverables and scope, the house of quality shown in Figure 2.2 was devised. The table connects the required deliverables shown on the left side of the table to important functional factors of scope shown on the upper row. Those required deliverables are additionally ranked by level of importance. The interior bulk of the table relays the correlation direction between these factors, a solid dot representing strong, hollow dot representing a medium, and a down arrow representing weak correlation. A similar metric is utilized on the roof of the house with positive and negative signs measuring the correlation between the functional requirements of the scope to one another. These features are connected diagonally with one another. The direction of improvement is added at the conclusion of the additional importance ratings as this allows for the team to best approach areas that require attention due to their high relation to the success of the project. The table shows the areas with the highest relative weight to be the most crucial to project success. This includes areas of accuracy, response time, functionality, and overall cost.

							+					
									+			
					+		+	+	+	+		
				+	+	+		-	-	+		
			+	+	+	+		+				
			Functional Requirements									
		Direction of Improvement	▲	▲	▲	□	□	▲	▼	▲	▲	▼
Relative Weight	Customer Importance	Requirements	Cost	Response Time	User Accuracy	Power Consumption	Product Size	Functionality	Admin Controls	BLE Accuracy	Shot Selection Quality	Audio Quality
2%	1	Ease of Use	●	●	●	▽	●	●	●	●	○	○
7%	4	Wait Time	●	●	●	○	▽	○	▽	●	▽	▽
5%	3	Navigation Accuracy	●	●	●	▽	▽	●	▽	●	▽	○
11%	6	Ideal Shot Selection	▽	▽	○	▽	○	●	▽	○	●	▽
15%	8	Differentiation of Balls	○	▽	●	▽	▽	○	▽	▽	●	▽
4%	2	Reliablity for Visually Impaired	○	●	●	▽	▽	●	●	●	●	●
13%	7	Affordable price	●	●	○	●	●	○	▽	●	○	○
9%	5	User Safety	▽	○	○	○	▽	○	●	○	▽	○
16%	9	System Display	●	▽	▽	●	●	○	○	○	○	▽
18%	10	Portability	○	▽	▽	○	●	▽	▽	▽	▽	○
		Importance Rating Sum (Importance x Relationship)	521.8	365.4	427.2	401.8	514.5	394.5	249.0	420	394.5	223.63
		Relative Weight	13%	9%	11%	10%	13%	10%	6%	11%	10%	6%

Correlations	
Positive	+
Negative	-
No Correlation	

Relationships		Weight
Strong	●	9
Medium	○	3
Weak	▽	1

Direction of Improvement	
Maximize	▲
Target	□
Minimize	▼

Figure 2.2 House of Quality Analysis

3. RESEARCH

This section of the paper covers the major topics of interest for VISION. From past projects to relevant technologies, this examination allows for technological solutions to be devised and properly informed for the project's design stage.

3.1 Similar Projects

Billiards Assistive Device for the Physically Challenged: A user assistive physical device was developed by the University of the West Indies to assist a user that was physically impaired and lost certain motor skills due to an accident. This mechanical device was aimed to improve grip strength, leading to improvements in overall performance.

Open Pool: This open source project is built around adding visual effects to the game of pool. By using computer vision powered by OpenCV, the computer can generate graphics by using the Unity game engine. The open source project gives step by step directions to set up both the hardware and software required for the project. The project requires a gray colored pool table, a Kinect Two for Windows, a computer with Windows OS, and a projector. The main areas of interest come from the computer vision code available. The main issue is that the project has not seen much maintenance since 2014. With all of the recent innovations in Computer Vision, it is unlikely the open source code can be used without major refactoring. However looking into the basic setup of the softwares OpenCV code may be of great benefit in our design strategy later on. Another feature of the project is code for detecting made shots, or “pocket detection” as the project named it. While they have released software for this feature, there is currently no hardware requiring us to fabricate the physical detection system ourselves.

3.2 Relevant Technologies

VISION does not aim to create a new form of technology, but rather incorporate many existing forms of technology into an innovative, inclusive system. The members of VISION have each become subject matter experts in their respective area of focus and have summarized their findings throughout the rest of this section.

3.2.1 Billiards Artificial Intelligence

3.2.1.1 Simulation Tools

The need for rapid simulation of games is needed to test the different shot selection approaches, as well as train our machine learning algorithms. These simulations will not encompass every shot parameter, but will let us make comparisons among the decision making models. Another effective strategy to model more realistic conditions will be to introduce noise to the simulations as well. By adding a normal random change to both shot power and angle we can better model a person.

Summary of Requirements:

- Latency of the shot selection algorithm does not exceed 5 seconds.
- Shot selection algorithm will produce a shot suggestion with a minimum specificity of 5 degree increments.
- Shot selection algorithm will produce a shot suggestion with a minimum specificity of 3 force levels.

Pool: This is the simulation software that was implemented in the paper "Deep Cue Learning: A Reinforcement Learning Agent for Playing Pool". The simulation software is further described in the reinforcement learning section below. This is an openly available project on GitHub.

Fastfiz: This is a version of the software Poolfiz and was used by the heuristic based model described below. This is an openly available project on GitHub.

Pooltool: This is a three dimensional simulation system for pool. The GUI operates very slowly, most likely because it is written in Python and handles 3D graphics. In order to be an effective option we would have to disconnect the shot selection algorithms from the graphical interface. The actual calculation of the shot however seems to take up a considerable amount of time as well. Dependency issues have been encountered while trying to use a special API for setting up physical simulations. In the documentation the author claims to not have put much work into the API thus far, and with little documentation, it may not be a very suitable choice. This is an openly available project on GitHub.

Ultimate Pool Simulator: A simulator written in Java. This simulation project has a built in GUI and multiplayer mode, allowing for each player to choose a shot. It was developed by a group of students for a class project and the physics would have to be evaluated extensively. This is an openly available project on GitHub.

Code Bullet Pool AI: This code has no documentation on its github page, the author created a youtube video for the project, but it is little help for setting up the project. It appears the code is written in an object oriented language such as Java or C++, but the .pde file extension makes it difficult to distinguish. The very limited documentation and no test cases lead me to believe this will be a difficult project to base our work on. This is an openly available project on GitHub.

Pool Genius: Pool genius features a GUI for displaying the shots that significantly slows down the program's performance. One shot took over 45 seconds to process, with only one ball remaining that was cut down to 10 seconds. The simulation is very slow and the overall shot selection process would likely need to be revised. This is most definitely not ideal for any sort of computations and would be much too slow for training against a model we make. While the shot selections are perfect, we may be able to tune down the performance on these in order to speed up computation. Another major consideration for this code is that there are no test cases currently available. Without these unit tests, it will be much harder to understand the code, as well as to make changes without breaking much of the functionality in unforeseen ways. This is an openly available project on GitHub.

PickPocket: This is a software developed by Micheal Smith, it is covered extensively in the section labeled Search Algorithms. The code is not openly available and we would have to request the source code, which is less preferable to an open source project with more documentation. The source code for this project would have to be obtained directly from the developer.

3.2.1.2 Simulation Tool Modifications

Shot Selection Algorithm Guidelines: We will be defining the shot selection algorithm as our way of deciding from what angle and with what force to hit the cue ball. For the purpose of our research, we will be looking at the table from only an overhead 2D perspective. This leaves out many important aspects of the game of pool, such as allowing for rotational momentum of the ball to change the shot. Our available simulation software makes it difficult to account for another axis. It would also be extremely difficult on any machine learning algorithms to add another axis for our output.

Limitations of Shot Selection Algorithms: The shot selection algorithm's usefulness is limited by human ability. The best shot may require perfect accuracy to hit correctly, and may be much more difficult than a safer alternative. That is why in most cases, the easiest shot is the best. For example, an algorithm may say there is a way for the player to make three balls at once, but it may require more precision than a human is capable of and may increase the risk of losing if a miss occurs. Another issue will be the communication from the algorithm to the person. Even if an accurate algorithm is produced, we must find a suitable way to communicate the power needed on the shot. Another issue is placing the user in the right location to hit the cue ball. Finally, the user may also strike the ball in an unpredicted way upon the vertical axis which our algorithm does not take into account. All of these factors lead to issues which must be taken into account for our algorithm.

Planned Simplifications: In order to simplify our model, we will be focusing only on a game in which only the horizontal angle of which the ball will be struck will be output by our model. This takes away the need to calculate spin on the ball, bringing down the complexity of shot selection immensely. We will also be only needing to come up with a shot selection algorithm for the solid color balls. This means that the algorithm does not need to interpret the positions of the opponents balls when coming up with a shot selection.

3.2.1.3 Different Implementations of Shot Selection Algorithms

Heuristic Model: This model is based on a research paper labeled “A Heuristic-Based Planner and Improved Controller for a Two-Layered Approach for the Game of Billiards” written by Jean-François Landry, Jean-Pierre Dussault, and Philippe Mahey (Landry et al.). This model used the Fastfiz simulator for simulating shots during testing. This model takes in 5 parameters : α horizontal offset from the ball's center; b vertical offset from the ball's center; θ angle of the cue stick in relation to the plan of the table; ϕ orientation of the cue stick; v initial speed given to the cue ball. The simulation tool Fastfiz is deterministic, so noise was added to the shot parameters to make results more realistic. An interesting heuristic found by the paper deals with safety shots, these are shots which are made to make it more difficult for the opponent to make a shot. These were determined to be impractical unless all other possible shot selections have a low probability of success. This is due to the difficulty of guessing what shot your opponent will take. The model in this paper uses a two layer approach, the name given to these two layers are the planner and the controller. Figure 3.1 below gives an overview of the planner architecture.

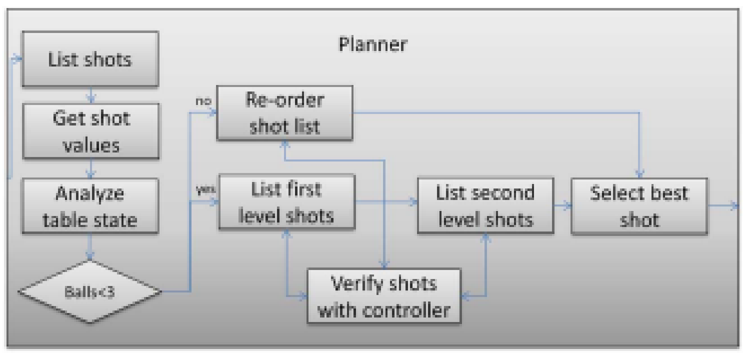


Figure 3.1 Shot Planner Diagram (Awaiting Permission from Jean-François Landry)

The high level planner uses several domain specific heuristics in order to narrow down the search space for the shot selection algorithm. At the beginning of a turn the planner determines which shots are possible, with this it creates a shot list made up of direct, combination, and indirect shots. It also lists all the pocket ball combinations. After this, the algorithm goes over the shot list and creates a difficulty value for every single shot on the list.

Another heuristic used by their algorithm is to always prefer shorter shots. The most successful approaches are the ones which require the cue ball to travel the least distance. This is due to the longer distance traveled creating for greater deviation from desired outcome as well as increased speed leading to more powerful and chaotic collisions. Another approach which was used was through the implementation of a k-means clustering algorithm which grouped the balls into different clusters. The reason that this method was added was to hit closest shots first, as those were generally the strongest shot choices. Another function we can take from this research is their formula for creating a function to penalize possible shots based on difficulty of the shot. For an easy shot, the direction is almost insignificant as long as the ball is tapped on a certain side. For more difficult shots, there is a much smaller area which the ball must be hit at and with a certain speed. An easy shot also allows for better positioning options, if there is a wider range of area on the ball you may hit to sink it into a hole, you then have more places to position the cue ball after the hit.

Reinforcement Learning Model: The reinforcement learning model is based upon trial and error in game-like situations. It is a machine learning algorithm implemented by using rewards and punishments. This model will find a locally optimal way to achieve a victory, or at least to maximize points. It is one of the most widely used models for creating an artificial intelligence system for games and therefore will serve well for pool. This will be much less time intensive than a supervised learning model. In a supervised learning model, the algorithm would imitate a human player. This would also create a model only as good as one of our team members, which is not at all optimal.

Assigning what constitutes a reward and punishment, as well as the relative weight of each is perhaps the most difficult part of designing a reinforcement learning system. We will try many different assignments, but some of the different rewards and punishments would be the following:

> Rewards: made ball (+1) or win game (+10)
> Punishments: made opponent ball (-1), scratch (-1), lose game by opponent(-5), or scratch on 8 ball (-10)

These systems often come up with unique methods that are not very intuitive. These shot selections may go against common knowledge and may be a poor way to teach newer pool players. Therefore we must thoroughly analyze this model once it is created to ensure that the shots selected are logical. On the other hand, this system may come up with better ways to cope with noise introduced to the system. A heuristic based model will work the same regardless of noise, but the reinforcement learning can learn to play with different levels of noise, thus modeling different skill levels of players. Exact thresholds for noise levels to model different levels of players would be arbitrary, but should be found by trial and error on our selection for the pool simulator. The source code for this project can be found on a publicly available GitHub repository as well.

We will be basing our research on a pool specific reinforcement learning model using a Markov Decision Making process with four different reinforcement learning algorithms: Q-Tablebased Q-Learning (Q-Table), Deep Q-Networks (DQN), and Asynchronous Advantage Actor-Critic (A3C) with continuous or discrete values (Liao et al.). This process is trained on the open source simulation project labeled “pool” in section 3.2.1.1.

Markov decision making process (MDP) is for modeling discrete decision or optimization problems where there is randomness and uncertainty in the problem. It can be represented mathematically as a 4-tuple (S, A, P, R) where:

> S is the set of states, called state space
> A is the set of actions, called the action space
> P is probability that action a in state s at time t will lead to state s’ at time t + 1
> R is the reward for transitioning from state s to s’ after action a

The sum total of different states may be finite or infinite, depending on the application. A game such as chess would have a finite number of different states to choose from, as well as discrete choices, making it a much easier decision making process. Pool on the other hand has a continuous range of actions as well as an infinite amount of possible states. The solution to an MDP is called a policy. This policy is a mapping from your current state to the preferred action in order to maximize rewards. This policy will form what is known as a markov chain, as the new state will also have a mapping to the next best state to achieve the best overall reward. A note about the markov chain is that it maps more than one probability, though the highest probability for reward will be selected in our case, there will be other paths that also offer reward from any state action pair. For the game of pool this will be very difficult to model. One such solution would be to choose the nodes of the MDP chain to be ball pocket pairs. This will however make it rather difficult to model shots that either hit the side of the pool table, or another ball before falling into the pocket. This method would also introduce much ambiguity in terms of angle and power, as there is a wide range of angles to result in any given ball pocket combination. Another option would be to discretize the power and angle of all shots. A discrete and finite pool action set would be as follows:

A = (Force, angle) = (F, $\boldsymbol{\theta}$)
F = [1, ... ,10]
$\boldsymbol{\theta}$ = [1, .. , 360]

A discrete and finite pool state set would be as follows:

S = [x_1, y_1, …, x_n, y_n]
Pool table is 127cm by 254cm, diameter of ball is 5.715cm, radius = 2.8575
Assuming y = [0, 253]
Assuming x = [0, 126]
Some values will be labeled as impossible to reach due to size of pool ball
* n is total number of remaining balls
* x_1, y_1 is the cue ball

Q-Learning: This is an algorithm to make the best selection in a MDP, otherwise known as a policy. This model learns the Q-values for every action and state pair. These Q-values are stored in a Q-table that maps actions on the horizontal axis and states on the vertical axis. The Q-learning method is applicable to a finite MDP. As mentioned previously simplifying the actions in the game of pool to a finite MDP can be difficult, the approach taken by the writers of the previously mentioned paper was to simplify the game of pool, similar to how was done above. The Q-learning algorithm works by referring to the Q-table and picking the action with the highest Q value for the given state, during training when the Q-table is empty the agent will make random actions in order to learn the different rewards for taking those actions, eventually filling in the Q-table. The main reason for this algorithm is to better understand delayed rewards in the system. There is a variable γ which represents the discount factor, when set to zero, the algorithm is myopic and simply picks the best current rewards (greedy algorithm), but by increasing this value, you find a path which gives higher long term rewards. An example Q-learning model is shown below:

$Q^{new}(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma\ (maxQ(s_t, a)) - Q(s_t, a_t))$
α is the learning rate
$Q(s_t, a_t)$ is the old value
$maxQ(s_t, a))$ is the best estimate of the optimal future value
γ is the discount factor

Deep Q Networks: This is used due to the fact that Q-Learning works well for a small number of state action pairs, but as this number grows, the algorithm becomes less efficient. In the case of a modeling pool, the number of table states is already so large, when paired with the vast amount of actions and the size of the Q-table grows too rapidly for most computers to handle. In the paper above, the Q-table for a simple two ball system was approximately 1.12 GB, and this number grows drastically as other balls are added onto the table. In order to combat this explosive growth of the Q-table size we will use a new learning algorithm. The total size of the state and actions pairs for our simple model would be on the order of $(\text{Action set} * \text{State set})^n$ where n is the number of balls. A deep Q network employs a neural network in order to come up with an approximation for the Q-learning algorithm. The input nodes for the neural network would be the current state of the table and the output nodes on the deep Q network represent every possible action. The value for that output node is the approximated Q-value. In our simplified case, 3600 output nodes is still significant, but the action set is much smaller than the state set and this is a preferred method in terms of space complexity. The total size of the model achieved in the paper was approximately 162 KB. The neural network consists of two hidden layers of 64 and 256 nodes respectively. Figures 3.2 and 3.3 below are two representations of what such a model may look like.

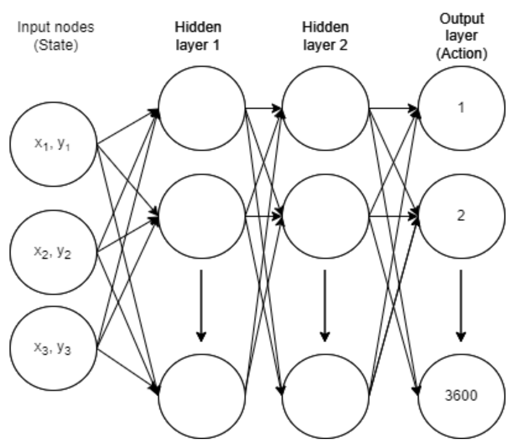


Figure 3.2 Neural Network Work for State Set with Three Balls

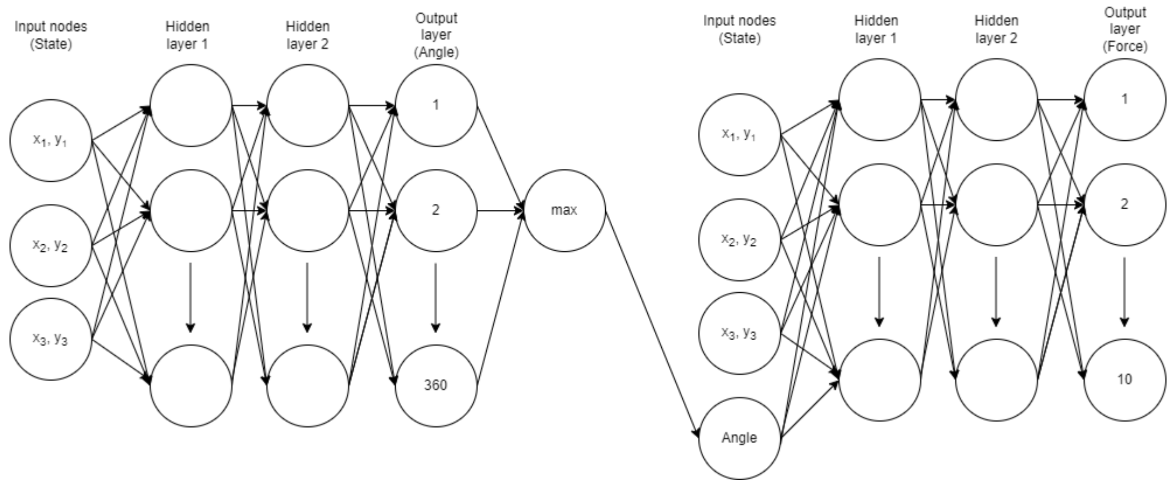


Figure 3.3 Neural Network for State Set with 3 Balls (Broken Into Two Networks)

Asynchronous Advantage Actor-Critic (A3C): This algorithm was developed by Google Deep Mind and first appeared in 2016. A3C implements several workers to gather information independently and asynchronously, then by using this information in a global network, the function value and policy may be estimated. While Deep Q-networks only use one environment and one agent in their training, AC3 uses several environments and agents. These agents act completely isolated from one another in their learning process, this allows for more diversified training and avoids local maximum optimizations. The other benefit of A3C is that it is useful for a problem with infinite space and infinite actions, meaning that it offers the most precise actions for any given space. This is done by breaking the model into an actor and a critic. The actor model takes in the environment and chooses the best possible action with its current data, while the critic model takes in the environment and acts as an evaluator for that choice.

The overall consensus put forth by the paper is that the A3C model was the most ideal model taking into account the training time and required space. The results for the models compared to the random baseline are not particularly impressive and would require refactoring to even get a usable amount of precision. Ultimately these algorithms do not seem to compare to the precision of search and heuristic based models. The benefits iof dealing with noise in the system may be a reason to attempt to build a model of our own.

Search Based Model: The research gathered for this section is for search algorithms in the game of pool. One major search based shot selection algorithm is known as “PickPocket” (Smith), this program would go on to win the first international computer billiards competition. One of the key points made is the inherent difficulty of using a search algorithm on a non deterministic and continuous set of outcomes. Search algorithms are a perfect way to choose the best move in a deterministic and discrete game such as chess, however, the difficulty is magnified in the game of pool. Another disadvantage is the considerable overhead required by the search algorithm to run a physics engine to determine the outcome of a given shot. This physics engine severely limits the breadth of the search tree. One such search algorithm suggested by the author is the Expectimax search algorithm.

The Expectimax search algorithm is a game theory algorithm that is a variation of the Minimax algorithm. While the Minimax algorithm expects the adversary to act optimally, the Expectimax algorithm expects the adversary to make non optimal decisions based somewhat on chance. The tree structure for this algorithm depends on nodes labeled as change nodes. These nodes present in the search tree represent points where the outcome is non-deterministic. An abstraction must be made in order to simplify the problem and use Expectimax. A pocketed shot effecting no other balls will result in a particular table state, while the missed shot can result in an infinite amount of different table states.

Another model which is brought up by the author is the Monte-Carlo simulation. This model is used in everything from modeling the card game poker to financial risk. The main purpose is to calculate probabilities of outcome when random intervention of variables is present. For the Monte-Carlo simulation, a number of samples or table states is calculated after each generated shot, each sample is a child node of the previous shot. This pattern trickles down to form a tree-like structure, with the score of each node being the average score of all the nodes children. The higher the number of samples, the more accurate the results. However the runtime increases exponentially as the number of samples are increased, therefore a proper balance must be found when using this simulation. When comparing this Monte-Carlo simulation to the previously mentioned Expectimax, you will see the main trade off is breadth vs. depth. The Monte-Carlo simulation has a much wider tree structure while the Expectimax is able to create a deeper tree structure.

3.2.1.4 Computation of Shot Selection Algorithm

The shot selection algorithm requires a system with high computational power for either a large search algorithm or heuristic algorithm. For a mathematically intensive machine learning algorithm, VISION would require a large computational resource for the training phase, but would require significantly less compute power thereafter. The use of a microcontroller will not be able to handle the large amount of processing needed. The options we may look for are either towards a microprocessor or a cloud computing solution.

Cloud Computing: In order to compute the function on a powerful machine and in a cost effective manner, one strong candidate is an Amazon Web Service product called a Lambda function. The lambda function allows you to run code on the cloud without having to manage the infrastructure. Instead of configuring and running a server on the cloud which is paid for based on time, you can instead use a lambda function which is paid for by usage. It has a strong use case for IoT backends and can be scaled quickly based on requirements. Amazon Lambda is currently on the free tier of AWS services and would be free to use for our small number of requests. There is also native support for Python, Java, Node.js, PowerShell and C# among others. This wide variety of options will allow us to implement almost any shot selection algorithm in the cloud. There is also a low amount of data being input into the lambda function as well as returned by the

Lambda function. This means that wireless communication bandwidth will not cause any large issues.

3.2.2 Computer Vision

3.2.2.1 Computer Vision Software Options

The computer vision portion of this project is the initial input to the entire system. An image will be captured from the camera and then processed by the selected computer vision algorithms. The chosen algorithms should be able to isolate the billiards table from the background, identify all of the billiard balls on the table, determine the position of all of the billiard balls on the table, and disregard all other objects on the table. The cue ball and eight ball, due to their importance in various billiard games, should also be distinguished from the other billiard balls on the table. The output of this subsystem should be the coordinates of all the billiards balls in play and a special identifier for the cue ball and eight ball.

Isolating the billiards table from the background can be accomplished by detecting the borders of the table and excluding all of the pixels outside of this border. The billiard balls can be identified by searching for circular contours, or outlines, in the image. The position of the billiard balls can be determined by utilizing the location of the circular contours previously found. All of the incorrectly-detected objects can be excluded by checking the size, shape, and color of all detected objects to ensure that only billiard balls are tracked. Finally, the cue ball can be distinguished from all of the other billiard balls by checking the color of the detected objects for a purely white object.

The requirements for this project are relatively common in computer vision and many of the current computer vision offerings are more than capable of the required functionality. The ideal software package for this project will require the least amount of computing power while ensuring high accuracy for detecting and locating the billiard balls. Furthermore, the ideal software will have a low latency to allow a user to play a game of billiards in a reasonable time. The requirements for the system are summarized below.

Summary of Requirements:
- System can locate up to 10 billiard balls
- System can differentiate between white, black, green, and blue billiard balls
- System can locate the balls in an (x,y) coordinate system with 15 pixels
- System can locate the six pockets in an (x,y) coordinate system within 15 pixels
- System latency does not exceed 5 seconds

OpenCV: OpenCV is a computer vision and machine learning library that provides C++, Python, Java, and MATLAB interfaces and is supported by all of the major operating systems. The library is open source and contains thousands of ready-to-use computer vision algorithms that have been used by many prominent companies like Google, Microsoft, Intel, IBM, Honda, and Toyota (OpenCV "About OpenCV"). OpenCV offers extensive support by providing forums, tutorials, courses, and detailed documentation.

OpenCV is written in optimized C++ code which allows for high-speed execution and a low software overhead.

SimpleCV: SimpleCV is an open-source framework developed by Sight Machine to easily develop computer vision projects. The framework combines various computer vision libraries, including OpenCV, and abstracts many of the low-level details away from the developer. SimpleCV prides itself on making computer vision easy and accessible to everyone (Sight Machine Inc.). The framework is written in Python and available on all major operating systems. SimpleCV has a larger software overhead because it is a framework rather than a single library. SimpleCV does not appear to be under development anymore, but still has a stable release available to download. The documentation, forums, and overall support of SimpleCV are much less useful when compared to the other computer vision offerings that are available.

TensorFlow: TensorFlow is an open-source machine learning platform made by Google to create, train, and implement designs. Tensorflow can be used with C, C++, Java, Go, or Python and supports many of the popular operating systems. Coca-Cola, Intel, Twitter, Airbnb, and other prominent companies utilize TensorFlow (TensorFlow "Why TensorFlow"). One of the main strengths of TensorFlow is the ability to train and deploy custom machine learning models. The software package also comes with many pre-trained models that can also be used.

Although TensorFlow was not designed specifically for computer vision, there is built-in support for computer vision applications. There is support for servers, IoT (Internet of Things) devices, and web devices. There is ample support for TensorFlow with many pre-trained models, datasets, blogs, forums, and tutorials readily available. Since TensorFlow is a collection of machine learning tools, it has a relatively high overhead when compared to some of the other computer vision offerings. The latency of this software package needs to be considered.

TensorFlow Lite: TensorFlow Lite is a specialized version of TensorFlow designed specifically for mobile and embedded devices. This software package is optimized for latency, privacy, connectivity, size, and power consumption (TensorFlow "TensorFlow Lite"). TensorFlow Lite can be used with Java, C++, Python, and other popular programming languages. It supports Linux and many common microcontroller operating systems. This software package requires little space on a microcontroller and incorporates hardware acceleration to boost performance and reduce latency. Similar to the standard TensorFlow, TensorFlow Lite was designed for machine learning but does support computer vision applications.

Nvidia Vision Programming Library (VPI): The Vision Programming Library (VPI) is a software library developed by Nvidia for computer vision and image processing applications. This library is optimized for performance on the Jetson Nano line of processors. The VPI supports both C++ and Python programming and is available on most major operating systems. The optimized algorithms in the VPI offer significantly better performance compared to many other computer vision tools and can be up to fifty times faster than similar software packages (NVIDIA Corporation). In addition to being highly efficient, the VPI can be used in conjunction with other popular computer vision tools. Most notably, the VPI easily integrates with OpenCV to quickly produce computer vision applications. The VPI is relatively new compared to some of the other computer vision tools and new versions are still currently being developed. There is not as much community support compared to OpenCV and TensorFlow, but Nvidia does offer a variety of tutorials and a forum where Nvidia developers frequently answer questions.

YOLOv3 (You Only Look Once): The You Only Look Once version 3 computer vision tool is an object detection algorithm that is built upon Keras and OpenCV. This algorithm was designed for fast real-time object detection, but can still be used to process images. The algorithm favors speed over accuracy and has a low accuracy for detecting small objects compared with other commonly used algorithms (Meel). Although newer versions of the YOLO algorithm have improved the accuracy, this software was not further pursued because of the low accuracy for small images.

Keras: Keras is a Python API designed to simplify the use of TensorFlow 2.0 for users. Keras abstracts away many of the low-level details associated with developing in Tensorflow while maintaining all of TensorFlow's benefits. The API prides itself on being simple, flexible, and powerful so that applications can be rapidly developed (Keras). Keras, like TensorFlow, was developed to be a machine learning tool and is used by NASA and YouTube. KerasCV is a subsection of Keras which supports many standard computer vision features such as image classification, object detection, and image manipulation. There is support for KerasCV in the form of guides, example code, forums, and a community supporting the software.

3.2.2.2 Computer Vision Preprocessing

OpenCV is the primary software being used for the computer vision needs of this project. Nvidia's VPI will be implemented if needed to improve the algorithm performance. OpenCV offers thousands of functions that perform a wide range of operations on images and videos. With so many possible options, it is important to narrow down the scope of OpenCV to a smaller number of relevant functions. This section discusses some of the necessary functions for image preprocessing that are needed for implementing various computer vision algorithms.

The initial input for the computer vision subsystem, and the entire system overall, is an image of the current state of the billiard table. The image preprocessing begins by converting the color space of the image from RGB to grayscale. Depending upon the selected algorithm, the image may also need to be thresholded. Thresholding of an image is essentially creating a binary image based on a threshold value. Finally, image filtering may also be needed to remove unwanted noise from the image or to prepare an image for subsequent algorithms. Some, or all, of these preprocessing steps, may be necessary before running object detection algorithms on the image.

Image Acquisition: The first step of all the needed algorithms is to capture the current state of the table. From this image, the position of the billiard balls will be extracted and later used by other subsystems of the project. OpenCV easily interfaces with any type of camera connected to the device on which the program is running. The selected webcam and how the webcam will be mounted are discussed in a future section. OpenCV will be used to control the webcam and capture the image when needed. OpenCV also easily allows for the captured image to be saved onto the device in which the program is running. This feature will help save images to later be used as output for the user.

Color Space Conversion RGB → Grayscale: Many of the computer vision algorithms that OpenCV implements require a grayscale image. By default, the input image is captured in RGB (red, green, blue) format. The RGB color format is how many images are displayed because it offers a wide range of possible coloring options to give the most accurate color representation of the image. Each pixel of the image will have an eight-bit red, green, and blue component typically displayed as a decimal value between 0-255. The combination of all of these color values is what defines the color of a pixel. While this large amount of color data is useful in displaying vibrant images, it is not helpful when trying to process an image.

To reduce the amount of computation needed, nearly all computer vision algorithms require that the image be converted from an RGB format to a grayscale format. This conversion allows for each pixel to be represented by one eight-bit value. A grayscale value of 0 corresponds to black while a grayscale value of 255 corresponds to white. With a grayscale conversion, all of the RGB-colored pixels of an image are mapped to a corresponding grayscale pixel. Although the color information is lost during a grayscale conversion, the information necessary to perform the computer vision algorithms is preserved. Specifically, the edges, regions, blobs, junctions, and other relevant information are maintained when an image is converted to grayscale (Breckon and Solomon 9-14).

The actual conversion of an RGB image to a grayscale image is simple in OpenCV. OpenCV allows for the conversion of color spaces with a call to the *cvtColor()* function. This function has many different predefined conversions that will allow for the input image to be converted to grayscale. One important detail to note is that the standard color format for OpenCV is BGR rather than RGB, a small modification will be needed to the function call when implementing the color conversion (OpenCV "Color Space Conversions"). The conversion of the initial input image from a color space to a grayscale space is lossy, meaning the initial image cannot be reconstructed easily. For this reason, the original input image must be saved so that it can be used in other parts of the project.

Image Thresholding: Some of the algorithms that OpenCV offers require an image to undergo thresholding before being processed. Specifically, algorithms that detect the edges of images utilize thresholding. Thresholding is a process to break an image into distinct regions of pixels to make images easier to process (Data Carpentry). In a sense, thresholding an image is converting it to binary because all of the pixels will be black or white. This type of image preprocessing is useful because distinct edges begin to form around features in the image which makes more complicated algorithms, like edge detection, possible.

One of the challenges of implementing image thresholding is determining what threshold value to use for an image. The threshold value will be used to determine which pixels are turned completely black and which are turned completely white. It can be difficult to determine an appropriate threshold value because the threshold will depend on the camera, lighting, and other factors that may not always be consistent. A common technique is to create a histogram of the intensities of the grayscale pixels as shown in figure 3.4 (Jayasekara et al. 530). Ideally, the histogram will have a clear distinction of values above and below the threshold. These histograms can be constructed in a variety of lighting conditions and an empirical value can be deduced from the findings.

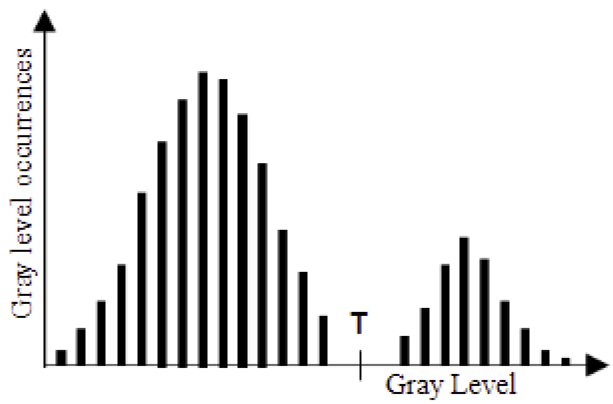


Figure 3.4: Ideal Distribution of Thresholding on Image (Permission Granted by Awantha Jayasiri)

Rather than empirically determining the threshold value, Otsu's method can be used for determining the optimal threshold value. Otsu's method works by iterating through possible threshold values and determining which threshold value gives the tightest clustering of black and white pixels (Muthukrishnan). Otsu's method tries many possible options and assigns values to the accuracy of the threshold, the highest value corresponds to the best threshold. While this approach does seem more accurate than the empirical approach, it will still be impacted by varying lighting conditions and where the billiard's table is located.

For both previously mentioned techniques, there is one threshold value used for the entire image. The technique of having one thresholding value is called global thresholding. Global thresholding faces challenges when the lighting and picture resolution are not uniform throughout an image. To mitigate these issues, adaptive thresholding can be used. Adaptive thresholding does not use a single global threshold value, but rather compares the grayscale values of neighborhoods of pixels to determine localized thresholds. This approach to thresholding accounts for lighting issues that may make one portion of an image darker than the rest. By using many threshold values, adaptive thresholding can produce much more accurate results and will typically outperform global thresholding techniques. An example of adaptive thresholding on objects of various colors and sizes is shown in figure 3.5 (Rosebrock).

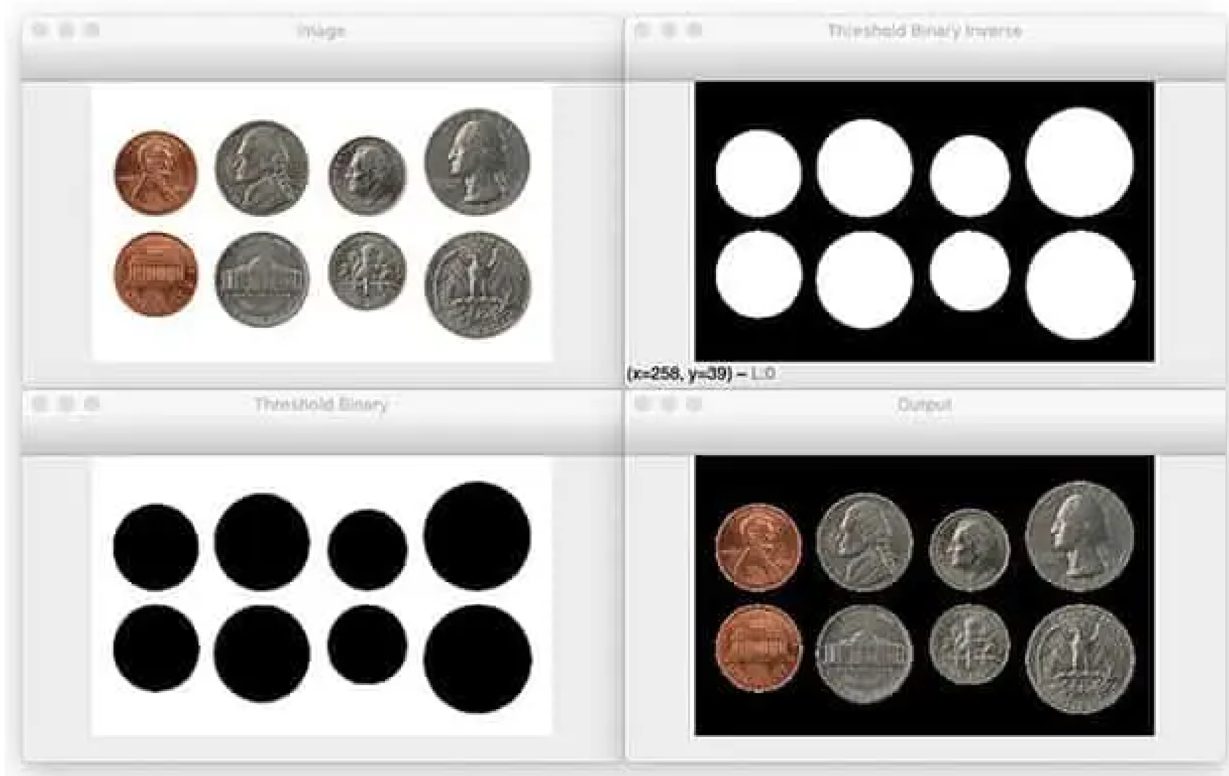


Figure 3.5: Image Thresholding to Isolate Region of Interest (Awaiting Permission from Adrian Rosebrock)

If image thresholding is needed for the selected computer vision algorithm, OpenCV supports all of the discussed thresholding techniques. More than likely, an adaptive thresholding algorithm would be used because of its better accuracy. OpenCV offers multiple different kinds of adaptive thresholding algorithms including adaptive mean thresholding and adaptive Gaussian thresholding. The specific type of adaptive thresholding used will depend on what computer vision algorithm is selected.

Image Filtering: Image filtering is the process of removing aspects of an image that are not desired to aid in processing the image. There are many different kinds of image filters available and they are most commonly used to remove noise, sharpen the edges, or blur the image together. These various types of filters are used for specific applications and help improve the quality of the final output. In general, image filtering occurs by looking at every pixel in the image and comparing it to all of its neighboring pixels through convolution. All of these pixels are then compared and altered based on the desired type of filtering.

One of the main applications for image filtering is noise removal. Noise, or unwanted additions to images, arises from many different factors related to how images are acquired. Many types of noise removal filters can be applied to images that come at a tradeoff of accuracy for computational complexity. Two of the simpler filters are the mean filter and the median filter. The mean filter is useful for removing uniform noise throughout an image but tends to worsen the image's overall clarity. The median filter is useful for removing salt-and-pepper noise, small regions of high-intensity noise, and is better at preserving the image clarity (Breckon and Solomon 90-94). A more complex filter is the Gaussian filter that can be used to remove noise, smooth an image, or prepare an image for edge detection. The Gaussian filter can be used for a wide range of applications because it allows the user to control a standard deviation parameter. Depending upon the value of this parameter, the filter can be used for different tasks.

Image filtering is also used to enhance an image before being used in an edge detection algorithm. Edge detection filters work by searching for regions of an image where there is a large amount of change occurring between pixels. Conceptually this represents a transition from one aspect of an image to another. Filters that are designed for edge detection locate these regions and amplify these transitions so that they are more easily seen during further processing. There are many different image filters available, OpenCV supports the Sobel, Scharr, and Laplacian filters (OpenCV "Image Gradients"). Overall, these filters are rather similar and most image processing algorithms will specify which filter is recommended to achieve the best results.

3.2.2.3 Computer Vision Algorithms

Once an image has undergone the necessary preprocessing, computer vision algorithms can be applied to extract the necessary information out of the image. This subsystem is responsible for isolating the billiards table from the background, identifying the billiard balls and their position, and differentiating the cue ball from the other billiard balls. The following section discusses image processing algorithms that can be used to achieve the computer vision goals of this project.

Canny Edge Detection: The Canny Edge Detection algorithm is a popular image processing technique that can be used to extract all of the edges from an image. This algorithm gained a lot of popularity because it was designed to exclude incorrect or misleading edges that previous algorithms tended to include. This algorithm can be useful for both isolating the table from the background as well as identifying the billiard balls. A sample image after undergoing canny edge detection is shown in figure 3.6 (BogoToBogo). The table itself will appear as the largest rectangular edge in the image and the billiard balls should be the only circular objects in the image. Using these characteristics, the table and billiard balls can be detected.

Figure 3.6: Canny Edge Detection on an Image (Awaiting Permission from BogoToBogo)

Canny Edge detection is a multi-step process that begins with filtering the image using a Gaussian filter to remove any present noise. A Sobel filter is then applied to find and magnify all of the discovered edges. The algorithm then checks all of the discovered edges and only allows the localized maximum pixels to pass to the next stage of the algorithm. This process ensures that the returned edges are the thinnest, most prominent edges in the image. The final step in the algorithm is another check of which edges should be returned and which edges should not. A hysteresis threshold is applied to the image. This is a threshold technique where two threshold values are used to identify only the strongest edge candidates and ignore the weaker edges (OpenCV "Canny Edge Detection").

This edge detection is appealing because it can offer a way to isolate the billiards table from the background of the input image. The border of the table is a nearly perfect rectangle and should be easily detected by this algorithm. Once the outer edge of the table has been detected, the space outside of the edge can be ignored. The image can be cropped or one of OpenCV's many functions can be used to mask everything outside of the table. Isolating the table will be beneficial because any further manipulation of the input image will have the background removed.

Template Matching: Template matching is a simple, but powerful algorithm for locating specific objects in an image. Template matching works by having a template, or sample image, of the object being searched for. The template begins in the upper left corner of the image and every pixel from the template is compared with every pixel in the input image. The template is then moved to the right by one pixel and the pixel comparison is done again. When the template reaches the end of a row, the template is moved down to the next row. This process, which is known as two-dimensional convolution, is repeated until the template has been compared in every possible location with the input image. Regions of the image that match the template will be assigned a high associativity value and regions that do not match the template will be assigned a low associativity value. The regions with the highest associativity values will be considered matches for the template (Adaptive-Vision).

The template image must be the same size as the object appearing in the input image. The template is being compared in every possible location in the input image. If the template is not the same size as the object in the input image, it is possible that the object will not be discovered or an incorrect object will be detected. Additionally, there are many ways to perform pixel comparisons. Different algorithms implement different pixel matching operations which can impact the algorithm's performance and accuracy. OpenCV implements six different operations which can all be used for template matching. The choice of which operation to use can be decided by trial and error with actual input images to determine which operation works best for the project.

One consideration when using template matching is if an RGB or grayscale image should be used for the input image. Most template matching algorithms, including the one supported in OpenCV, allow for both colored and grayscale inputs to be used. The benefit of using colored input images is that the algorithm will be able to better detect matches of a specific color. The increased matching ability is because there will be significantly more pixel values to compare the template image with. The drawback to using colored input images is that the algorithm becomes more computationally complex because now each pixel has a red, green, and blue component to compare. When using a colored input, the algorithm is essentially run three times, once for each color channel, and the results are averaged together for each pixel (OpenCV "Object Detection").

Template matching would be beneficial to use when trying to identify and localize the billiard balls in the input image. The maximum number and possible colors of the billiard balls being used will be known. Each of the billiard balls can have its own template image and the algorithm can be run for each possible billiard ball. There will need to be some type of confirmation that the object detected by each iteration of the algorithm found the correct billiard ball because some of the balls will not be on the billiards table. This approach also may be too computationally complex and lead to high latency. If the algorithm is run for each possible billiard ball using a colored input image, there will be a lot of intensive computation every time the state of the billiards table changes.

Suzuki's Algorithm (Finding All Contours): Contours in image processing are the lines that join all of the points along the border of some shape or object. Contours can be thought of as the outline of an object that is made between the object and the background. This idea is useful because the expected contours of the billiard balls and the billiard table can be used to detect these objects. An algorithm that finds all of the contours present in an image can be run, and the contours that are found can be filtered to extract only the desired contours.

Suzuki's algorithm, which is implemented by OpenCV, works by traversing the input image pixel by pixel from the top left to the bottom right. The algorithm works by comparing the value of a pixel to the values of the surrounding pixels. For many implementations of this algorithm, a binary image is required. As each pixel is examined, it is assigned a value that can be used to determine if an outer border, hole border, or neither has been discovered (Kang and Atul). These results can then be used to determine what contours exist in an image.

Finding all of the contours in an image is a useful feature, but contours that are not desired will also be found. To be able to successfully implement this algorithm, all of the contours that are found will need to be filtered. Only the contours of the billiard balls and billiard table should be returned from the computer vision system. The main application of this algorithm would be to detect and localize the billiard balls and the billiard table. For this reason, any contour that is not a quadrilateral or a circle can be ignored. It is possible to approximate all of the contours to common geometric shapes by using the *approxPolyDP()* function in OpenCV. The number of edges present in the contours can then be compared to the expected values. The contour of the billiards table should have four edges and the contours of the billiard balls should have more than eight edges (more than eight edges represent a circular shape) (Authentise).

Further filtering can also be implemented to ensure that the contours that are found are also of the expected size. While the exact size of the billiard balls and billiard table cannot be determined until the testing begins, the concept of relative size still holds. Once the billiards table and camera have been acquired, the algorithm can be implemented and the area of contours of interest can be recorded. A minimum and maximum size for the billiard balls and billiard table can be determined so that is unlikely incorrect contours are reported. OpenCV supports finding the area of a contour as well as contour highlighting. Contour highlighting can be used to view what contours are being discovered and adjust the filtering portion of the algorithm as needed.

Suzuki's algorithm would be useful in locating the billiard balls and the billiard table from the input image. Although this algorithm will likely return contours that are not wanted, OpenCV offers many ways to sort through the contours and extract only the relevant objects. This approach allows for a user to place tight guidelines on what objects are detected but will require testing and refinement to ensure that the filtering parameters are correct and reliable.

Hough Circle Transform: The Hough Circle Transform is a computer vision algorithm that can be used to detect all of the circles in an image. This algorithm allows for circles of a certain radius to be discovered in an image. All other shapes and any circles that have a radius that is either too big or too small will be ignored by the algorithm. This algorithm is relatively accurate and can ignore most shapes that do not fit the search criteria.

The Hough Circle Transform works by utilizing the characteristics of circles. All circles will have a center and some radius that is fixed for any point on the circle. Consider some arbitrary circle *c* with radius *r*. This algorithm works by traversing the perimeter of circle c and essentially drawing a circle, still with radius *r*, at every point along the perimeter. There will be one point of intersection in which all of the circles that are drawn while traversing circle *c* overlap with each other (ImageJ). This point will be the center of circle *c*. Every intersection is awarded a point and the center of the circle will have a very high point concentration compared to the surrounding pixels. The algorithm uses the point concentration relative to the neighboring pixels to determine if there is a circle present.

Many implementations of the algorithm require an outline of the objects being searched for in a binary image format. This requirement can easily be met by using the Canny Edge Detection algorithm discussed previously. The outlines in the image are what form the perimeter to be traversed by the Hough Circle algorithm. By using the outline of the objects it is also possible to detect overlapping or touching circles as well like shown in figure 3.7 (Sinha). If two circles are overlapped, the perimeter will form a shape that looks similar to the number eight. As the transform traverses the perimeter, it is often able to detect both circles, assuming they are of the same radius. This feature is because two centers will be found that have high concentrations of overlapping pixels compared to the rest of the image. The image below depicts when two overlapping circles of the same radius are detected.

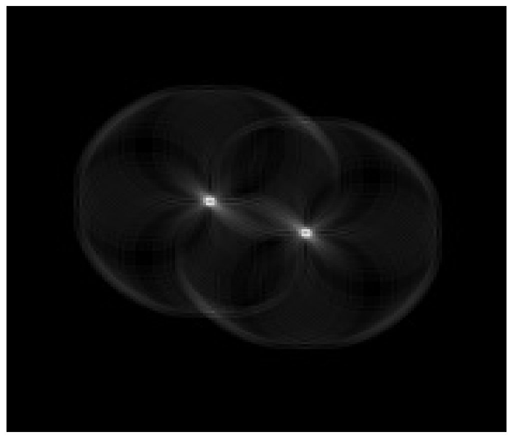


Figure 3.7: Detection of Overlapping Circles (Awaiting Permission from Utkarsh Sinha)

Similar to Suzuki's algorithm, unwanted circles may be found by the algorithm. Filtering of the circles found by the algorithm may be needed to ensure that only the billiard balls are detected. Fortunately, OpenCV's implementation of the algorithm allows for the minimum and maximum radius to be specified. The optimal values for these thresholds will need to be determined experimentally. Further filtering can be done by checking the color of the discovered circles to ensure that it is an expected color.

The main application of the Hough Circle Transform would be identifying and locating the billiard balls in the image. This task is one of the main goals of the computer vision subsection, and this transform looks very promising to accomplish the goal. One other related application would be identifying the pockets on the billiards table. Although this algorithm will not be able to isolate the table itself, the algorithm should be able to detect the pockets of the billiards table. Although the pockets do not form perfect circles, they are relatively circular and the algorithm should be able to detect them with only minor modifications. Detecting the pockets would help localize the coordinates of the billiard balls.

Douglas-Peucker Algorithm (Contour Approximation): The Douglas-Peucker algorithm is used to approximate complex contours into simpler contours. This algorithm essentially takes a detailed contour and simplifies it into a geometric shape such as a triangle, squa re, or similar shape. An examples of the contour simplification is shown in figure 3.8 (OpenCV "Contour Features"). The amount of simplification applied to a contour typically depends on an input parameter, epsilon, as well as if the expected simplified contour should be a closed shape. The algorithm works by determining the starting and ending points of the contour. The edges between these two points are what will be simplified. The algorithm uses the epsilon value to compare the distance from each point on the contour to a reference line. Points that become smaller than the epsilon value are discarded and those that are larger than the epsilon value are kept (Lee).

The value of epsilon used in this algorithm is crucial to what type of contour will be detected in the image. In the figure below, the leftmost image is the input image. The green outline in the middle image shows the discovered contour for an epsilon value of 10%. The green outline in the rightmost image shows the extracted contour for an epsilon value of 1%. As the value of epsilon decreases, the more tightly the modified contour will resemble the actual contour.

Figure 3.8: Epsilon Value on Algorithm Output (Awaiting Permission from OpenCV)

Like many of the other algorithms discussed, the Douglas-Peucker Algorithm requires a binary image as input. Furthermore, the algorithm requires that all of the contours in the image have already been discovered. These requirements can be accomplished by using previously discussed functionalities supported by OpenCV such as thresholding and the Canny Edge Detection algorithm. The value of epsilon to use will need to be determined experimentally, but will likely be relatively high because the billiards table is nearly a rectangle.

This algorithm will be useful for extracting the billiards table from the input image. More specifically, this algorithm would be used for drawing a rectangular contour around the playing area of the billiards table. The playable area is a nearly perfect rectangle except for the six pockets. If the pockets can be ignored, by the use of this algorithm, a rectangular contour can isolate the playable area. Once the playable area has been isolated, it will be much easier to localize the billiard balls as well as the pockets.

3.2.3 Visual Impairment Assistive Technology

Visual impairment is not something new to humanity. Individuals who suffer from this setback have learned to adapt to the setback for generations, but only in the last century has technology rapidly accelerated this progress to such an extent that life can gradually approach normality for those affected by visual impairment. To best guide this project's goal of assisting impaired billiard players, several previously designed assistive technologies are examined.

The goal of this project is to have a complete guide for users to be able to play billiards. From interfacing with system controls to navigation around the table to the AI generated desired shot, we want the system to be able to be utilized with little to no help from exterior users to the impaired player. The desired result is therefore a start-to-finish setup where the user can locate the table, select various settings of the game/devices, and conclude with executing desired shots within the game. What is examined for these compatible deliverables is a user interface that is able to be navigated either solely by touch or sound and a guidance system that utilizes sound or sensation to prompt a user toward a desired direction or specific location. There are several cases that are outside of the scope of assistance in this project. These include setting up the preliminary orientation of the balls, location of the user's cue, and obstacle avoidance.

With the constraints of the assistive technology outlined, two primary interfaces must be examined for the assistive technology deployed in the project: guidance and communication interfaces. The user interface seeks to communicate in ways that enhance the ability for mild impairments to be able to see options - an easy to use, simple, and observable UI, and that can be deployed in the case of a fully impaired user. Screen readers and voice technology have become commonplace in much of the technology that is now deployed that will read out what is displayed and highlighted on a screen. Within a similar realm, screen magnification softwares are deployed across devices for users that may have mild visual impairment ("Assistive Technology for the Blind (AT)"). System settings that perform these actions can be a verbal and visual enhancement for a user when navigating a settings page, attempting to start a game, or understanding the layout of a table and specifying the outlined shot. Additionally, braille keyboards and critical buttons are an age-old communication method that can be deployed for the completely blind to communicate with a device when fully powered off.

In terms of user guidance, the project will require methodology that tracks the user and deploys instructions that will locate the user at a desired destination for the optimal shot. Although the project is focused on a specific focus, previously designed technology validates possible options for the desired system and can give insight into how the project's goals can be realized. Localization algorithms such as visual-inertial odometry (VIO) utilize smart phones with a combination of computer vision software and the device's internal measurement units (IMUs) to understand a user's orientation and their current trajectory. Previous research in this realm utilized common benchmarks within a predetermined area to give a relative understanding of their location in a 2-D space. Given the inputs from the camera and the acceleration recorded within the IMU, the device could garner an accurate understanding of the user's location and guide them accordingly through an area that is previously known (Fusco and Coughlan).

Other research breaks down closer to the deployed microcontroller level of localization. A proposed system from Middle Technical University utilizes a IoT machine-to-machine protocol called ZigBee to localize a user relative to several anchor nodes in a room, and an RFID is used to recognize the interior the user has entered (shown in Figure 3.9). The system also scales for wider navigational purposes by using GPS to localize the outdoor position of the user, and alternates between the two depending on location . ("Localization Techniques for Blind People in Outdoor/Indoor Environments: Review").

Some visually impaired assistive systems rely less on user localization and more on environmental surroundings. The Sanjivani College of Engineering explored a command based audio input and output assistant that utilized camera inputs and a chatbot functionality to relay meaningful information to the user of their surroundings. The system consisted of a camera, headphones, and a microphone with several core functions including face and emotion recognition, image captioning, object detection, reading, and interfacing directly with a personal assistant bot. This system was fully local to the user and navigated based on user pronounced commands and the inputs given by surrounding by use of python APIs and CV software and then relayed meaningful responses by means of Google's text to speech platform gTTS ("Smart Guidance System for Blind with Wireless Voice Playback").

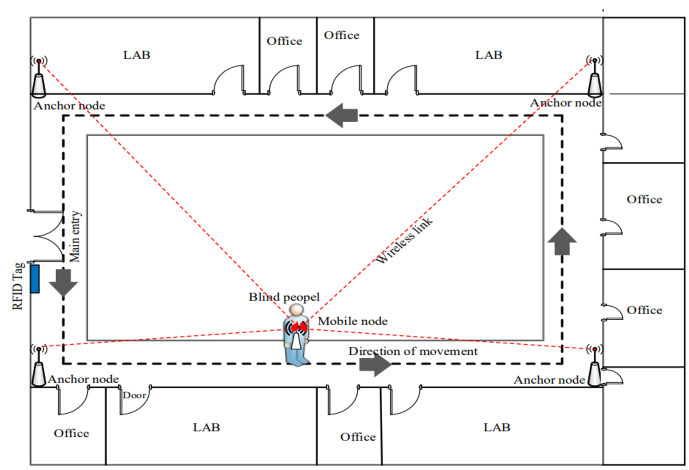


Figure 3.9: Previous System Indoor Localization Design ("Smart Guidance System for Blind with Wireless Voice Playback") (Awaiting Permission from Sadik Ghargan)

Another smart guidance system relies on several different approaches for determining critical obstacles, determining important events, and delivers audio feedback messages to the user. The Sri Sairam Engineering College developed a system deploying a voice feedback system for navigation that utilized an ultrasonic sensor to safely avoid objects and utilized a MEMS accelerometer for the purpose of understanding the user's dynamic location in a 3-D space. In addition to an accurate portrayal of the user's location, the static location was also understood using this accelerometer and a message was sent to points of contacts in the possible case of an emergency occurring. GPS was used to record the known location of the system and user, and would communicate the location in case of emergency ("Smart Guidance System for Blind with Wireless Voice Playback").

As audio assistive systems is a widely deployed approach, the subsystems for many past projects is a key point of interest for how to read in information and the different data points they focus on. Sensors for navigation can span many technologies. Deploying technologies in conjunction with one another enhances the full picture of the scope of the user's surroundings. For instance, many systems focus on deploying the commonly conjoined ultrasonic sensors and RFID readers to navigate premapped areas and avoid obstacles throughout those regions ("Audio guidance system for blind"). On the other hand, technology such as LiDAR has shown to be viable in the past for the visually impaired ("Voice Navigation Based guiding Device for Visually Impaired People") and can be viewed as a more independent sensor system that is powerful in the full picture it can paint for a system software.

Previous iterations of visual impairment assistive technology lay a good framework for how to best guide users in the scope of navigating a billiards game. User guidance, control, and safety are the primary goals of the system. Emphasizing these by enhancing the ease of use can be best improved by seeing where these projects examined shortcomings and seeing where they can best be improved upon. The following sections research some of the required technology for user interaction to be possible in greater detail.

3.2.4 User Localization

This section describes different technologies or avenues that can be explored for user detection, including but not limited to visually impaired users, technology that could be used in further sections when considering determining the path for the user to the object of interest. The current scope of research is to find how to implement three different features for the user. Further sections will describe which features will be implemented and in which way each of the features will be implemented. This section outlines the process of how to navigate a user to the billiards table, describes how to detect any obstacles in the user's path to the table or around the table, and describes how to navigate the user around the table to the right position and orient the user is the direction needed to make a shot based on the shot selection algorithm's output.

To do any of the navigation accurately and safely, a proper localization mechanism must be deployed so the user can receive instructions that correspond with their location and heading in real time. Several variables are considered and must be prioritized accordingly for end design selection across various sensors and the corresponding algorithms that can be deployed with them. Variables to consider for each method of sensing would revolve around: accuracy, calibration techniques, computational bandwidth, resolution, range, outstanding environmental factors, cost, ease of user integration, scale, materials required, and the method of sensing (i.e. proximity, motion, image, etc.) (Into Robotics).

Hence, here, we examine different technologies, such as RFID and infrared/ultrasonic sensing and ultimately summarize our options and determine what sensors or sensor technologies we go for and in which matter they will be interfaced in our final physical design described in later sections.

Summary of Requirements:

- Latency of the user localization does not exceed 10 seconds
- Accuracy of the user localization is within 1 foot of the true location
- Localization should work independently of the surroundings

3.2.4.1 RFID And Bluetooth

<u>*RFID:*</u> RFID (Radio Frequency Identification) is a form of wireless communication using radio frequency (RF) waves to identify objects uniquely. RFID systems consist of scanning antennas, transponders, and transceivers. Transceivers and antennas can be combined in an RFID reader. Transponders are typically RFID tags. In practice, mobile or physically mounted RFID readers would be located within the region of application transmitting waves within the RF spectrum. The waves are picked up by the RFID tag(s) which will send the signal back to the antenna portion of the RFID reader, a signal which will be turned into data and positioning information. The range of applications depends on the type of RFID readers and tags and the RFID frequency of operation. Table 3.1 summarizes the different types of RFID systems based on the frequencies of operations.

RFID System	Frequency Range	Common Frequency	Operation Range	RFID Tag Pricing
Low-Frequency (LF) RFID Systems	30KHz - 300KHz	125KHz - 134KHz	≤ 10cm	\$0.5 - \$5
High-Frequency (HF) RFID Systems	3MHz - 30MHz	13.56MHz	≤ 30cm	\$0.20 - \$10.00
Ultra High Frequency (UHF) RFID Systems	300MHz - 3GHz	433MHz, 860MHz - 960MHz	≤ 100m	Depends on Active vs Passive Tags

Table 3.1: Comparison of RFID Technologies

These systems not only determine the range of frequency and application but also narrow down our options for tags and readers given that in most instances, the specific type (LF, HF, UHF) of RFID tag can only be read by the same type of RFID reader. LF and HF systems are typically used for close contact applications due to their short range of detection and limited speed, as in ticketing systems, payments, or access control.

There are three relevant use cases for VISION. If used for constant user detection in a wide room (approximately 10m), Ultra High-Frequency RFID systems are our only option to make sure that our user of interest, wearing an RFID tag, will be detected by our reader. For obstacle detection, either Ultra High-Frequency or High-Frequency RFID systems will work since the range of detection will only need to be in a very narrow range of about 30cm. However, considering the unpredictability of the locations of the obstacles themselves, RFID might not be a suitable solution for direct obstacle detection unless the obstacles will always be the same and located in the same spots at all times. In this case we could detect all fixed obstacles but would need another system to detect any other obstacles in the direct vicinity of the user on their way to the table or navigating around the table. For our third use case, we would have to rely on either Ultra High-Frequency or High-Frequency systems to locate the user from the edge of the pool table depending on how far away from the table the user is located. 30cm could be sufficient in some cases, but Ultra High-Frequency systems would be a more reliable approach in this case. If this solution is used, the applicability, availability and price of either one of these two solutions will need to be further evaluated. Now that our choice of the RFID system is determined, the next step will be selecting which RFID readers and tags would be suitable for our application.

RFID Tags: As earlier mentioned, RFID tags consist of the transceiver, an antenna capable of receiving and transmitting signals, but also the RFID chip, which stores the tag's ID. For UHF RFID systems, there are three different types of RFID tags: passive (solely powered by electromagnetic waves), active (powered by a battery), and battery-assisted (combination of active and passive). The latter two allow achieving much longer ranges, at the cost of a much higher price per tag. Other considerations in selecting the proper tag are described below:

- Size: The larger the size, the longer the read range. However, this size is limited by the size of the object being tagged, in this case, our physical design or other objects, which incorporates the tag.

- Alignment and orientation: Ideally, the tag should be aligned in the same plane as the RFID reader to maximize the absorption of RF energy. Testing, if needed at this range, will need to be done to find the proper alignment for the reader and the tag. Additional readers may be positioned in the room of interest if needed to minimize issues arising from this.

- Application-based type: Depending on the vendor, RFID tags are broken down into different categories including hard tags, wet and dry inlays (paper tags with or without adhesive), sensor tags, high-temperature tags, and embeddable tags, among others.

- Resistance to impact, vibrations extreme temperatures, UV, dust, or other chemicals

For this specific application, wet or dry inlays will be the best option considering the cost and the fact that there is no necessity in a bigger or more complex design for our tags. Singular tags or multiple tags can be placeheyitssolano@gmail.comd upon our physical design worn by the user, on different sections of the table, or on objects to be detected, depending on the size of the said obstacle (for fixed obstacles in the room). An apt example would be Avery Dennison's AD-172u7 inlays which feature a 22 x 12.5 mm antenna designed to operate at around 860-930 MHz, each inlay factory locked with a unique 48-bit identification number while sitting at a total pitch of less than 2 inches. ("UHF RFID Inlay: AD-172u7 - Avery Dennison"). The AD-172u7 is shown below in figure 3.10.

Figure 3.10: AD-172u7 UHF RFID Tag and Inlay (Permission Granted from RBIS Americas)

RFID Reader: As earlier mentioned, RFID readers are responsible for sending signals to and receiving signals back from RFID tags. The two main types of RFID readers are either fixed or mobile, further subdivided based on the RFID system in play. Moreover, RFID readers can be further divided based on connectivity options (Wi-Fi, Bluetooth, Serial, USB, LAN), number of antenna ports, power, and processing options. RFID antennas are typically also necessary in addition to RFID readers, since they help convert the RFID reader signal into RF waves that can be picked up by the tags. The antenna will have to be in the same plane or polarity and orientation as the reader to superimpose instead of nullifying their actions. RFID antennas could also be used to facilitate communication between the antenna and the RFID reader. If used for obstacle detection or for navigating the user around the table towards the optimal shooting position, the RFID reader will be incorporated into our physical design, allowing our system to detect the RFID tags placed on different obstacles and use that information to navigate the user around the room. If used for moving the user around the table, the RFID reader would need to be able to distinguish tags that may be placed in very close location since the user holding a tag might have to be in close contact with different tags placed around the table (if any). If this solution is implemented, the choice of RFID reader will need to take this issue into account. If used for user detection, the RFID reader would need to be positioned in the middle of our room containing the pool table, allowing its area of detection to pan out as much as possible in the room.

RFID Applications: The most accurate way-finding technologies used for visually impaired individuals these days rely on RFID technology. Despite how relatively inexpensive RFID tags (mainly inlays) are, the biggest cost in these come from RFID readers whose cost vary from around $200 to ten times that or more. Justifying the use of RFID and RFID readers for user identification would involve using RFID for user positioning as well. Other technologies rely on HF RFID systems and make use of NFC (Near Field Communications) which does not need a separate reader, smartphones can serve as a reader for NFC, but are limited to about a few centimeters and typically operate on identifying one tag at a time making them unsuitable for identification or way-finding of visually impaired individuals. Another justification for the use of RFID would be with multiple user detections, where a system of RFID detectors or readers can be positioned at different points in a building identifying and detecting the positions of users with specific RFID tags. With these considerations, we could opt for the cheapest possible UHF RFID reader compatible with our tag selections from earlier.

Related to our current application, RFID tags and readers have been used in different ways for navigation in buildings or in smaller cases single rooms. Most of these applications rely on having a predetermined network of RFID tags, implemented as checkpoints, at specific locations such as doors, corners or windows for building applications and allowing the user, with the reader, to walk around and be given directions whenever their reader detects another tag. Another application, based in a single room, positions the RFID tags mounted on the ground such that the tags are separated by 0.682m, equidistant from each other, to avoid collision from the reader. Every time the tag is detected, its unique ID is verified by the microcontroller allowing an algorithm to determine the closest path to the pool table's ID and making sure that the user is still following the previously set path. In the possibility that there is an obstacle in the path the user follows from tag to tag to reach their destination, this application also introduces the use of sensors for obstacle collision. If an obstacle was detected along the path, the software generates an alternative path to the destination for the user to avoid the obstacle.

Related to our third use case, user navigation around the table, there are cases were RFID tags are being used in the dining industry allowing waiters to find guests at the right table based on the specific location returned by an RFID tag preemptively given to them. In a similar way, we should be able to differentiate different seating positions that would correspond to a grid breakdown of what the pool table looks like and know exactly at which position, that is at which RFID tag the user is currently located at. Alternatively, we could simply detect the user's position using their RFID tag and use different ways to relate that positioning to the targeted position determined by the algorithm without using additional RFID tags to confirm that the targeted position has indeed been reached.

Bluetooth Low Energy (BLE): A considerable alternative to using RFID technology would be relying on Bluetooth Low Energy systems to achieve the same functionalities described earlier. BLE is a radio frequency technology for wireless communication that can be used to detect and track the position of different objects or people. They operate in a range similar to regular Bluetooth (about 2.400–2.4835 GHz) comparable to Ultra High Frequency RFID systems. The LE portion in the name refers to its low power and current consumption (0.01 to 0.5W versus 1W reference for regular Bluetooth and <15mA of current consumption).

BLE localization typically uses BLE beacons placed at specific points in our area of interest, providing information on the specific location of different objects in the area of interest or breaking down the overall area into specific grid locations. These beacons are small, versatile Bluetooth transmitters which broadcast signals at regular intervals. These signals that can be detected by wireless devices such as BLE enabled smartphones. This describes a major advantage of BLE versus RFID. The overly expensive RFID readers can be replaced by regular smartphones that natively support BLE. However, the major issue described when using RFID tags in close proximity would still be an issue for this application. The efficiency of this technology will differ when you take into account different factors like the beacons not transmitting information to the reader synchronously while the user is in motion, or the reader struggling to detect closely placed beacons.

BLE Localization Techniques: Different localization techniques also come into play depending on the application or use case of these beacons. The simplest one would be localization based on the random detection of transmitters or beacons. In this technique, the position is based on which beacon provides the strongest signal back to the reader. Similar to RFID tags, we would need to store information about the different beacons to determine the location of the closest beacon to our user. The strongest signal would be calculated by a combination of three different values. The first one is an RSSI (Received Signal Strength Indicator) value, which indicates how strong the received signal reaching the mobile device when the beacon is detected by our device or reader. In addition to this, we have to consider that different beacons would broadcast their signal at different transmission powers TX. A combination of the RSSI value and the TX power value must be used when estimating the distance to the beacon. The TX power value is a factory-calibrated, read-only constant that indicates the strength of the signal measured at 1m from the device. Another consideration is a constant, say N, which represents the path loss index and is dependent on the localization environment. Some different values of N are: 1.4–1.9 for corridors, 2 for large open rooms, 3 for furnished rooms, 4 for densely furnished rooms, and 5 between different floors. Using these values, we can calculate the distance based on the following formula:

$$d = 10^{(TX - RSSI)/10n}$$

The major issue with this approach is that this localization technique varies greatly depending on the area in which it is been used (denoted by the range of values for N). This is only relevant if we are detecting the user in the room with beacons laid out throughout the room and are trying to bring the user to the table where another beacon is being placed. Single measurements from the different beacons could consider one as the strongest signal at a particular moment, but measuring it again would lead to another beacon being deemed the strongest signal. A solution for this could be implementing an algorithm that uses a moving average over a period of time. This could introduce a longer time for detection depending on the scanning interval and scanning duration used for the algorithm. We could increase the frequency of detection while reducing the scanning interval, but this would contradict the whole point of having a diverse average to get the most accurate outcome. Research done with beacons closely packed under this technique has also shown that when placing them close together - for instance at 25 cm - the accuracy of detection is below 50%, detecting the wrong beacon or transmitter more than half of the time. (Cannizzaro)

If we only focus on detecting the user around the table, we might only need one beacon on the user and have to detect its position using a reader or mobile device placed in a strategic location. Alternatively, as described before, we could have beacons around the table and have the user be the reader, but this would still raise the issue of having readers interfering and reporting false measurements from the beacons being in close proximity. Revisiting the first solution, another concern that would have to be investigated in our physical design, is the effect of obstacles around the user. The RSSI values are affected depending on different obstacles or objects in their vicinity. Depending on the density of the obstacles, it has been shown that some detections from the beacons might be lost, and the RSSI values may have a range of error of about ∓5 which in a narrow area like the pool table could lead to faulty measurements of where the user is accurately located.

Another more accurate, but complex, localization technique is trilateration. Trilateration determines the location of the object or person of interest by using three strategically placed beacons. The beacons draw out a circle, with the beacon at the center of the circle, in their location, and the intersection of the circumferences determines the exact position of the object of interest. In details, data from each individual beacon allows us to have a general idea of where the object is located within the beacon's drawn out circle. This location comes with a great range of error. The location of the object due to the second beacon will allow us to remove some of this error by placing the object in the overlap of those two drawn out circles, reducing the plausible region where the object would be located. The third beacon would in turn reduce this area to a single point, giving us the exact location of the object. The horizontal and vertical positions of the objects are then determined based on the radii of the said circles and the distance between the beacons. Those distances are calculated based on RSSI and TX as earlier described. A simple trilateration example is shown below in figure 3.11.

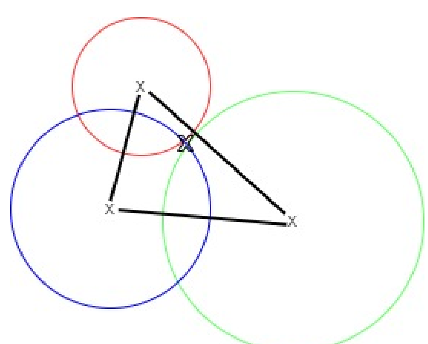


Figure 3.11: Simplified Model of Trilateration

Regardless of the method used to determine the exact position of the user, it might be worth finding ways to minimize the error incurred in the RSSI measurements, which is the basis of the whole process. The moving average described for successive measurements earlier is one of those but can be improved to smooth the RSSI values even more. Different models, such as exponential moving average, or weighted moving average, could be introduced such that the RSSI value is not just a simple average of the previous values, but gives greater importance to newer values versus older values. This would help with cases where the user might be in constant motion around the table or in the room. Consider $RSSI_n$ to be the current RSSI measurement, $RSSI_{smoothed}$ is the smoothed calculated value and α is a number between zero and one. A smoothing model is shown below: (Ramirez and Chien-Yi Huang)

$$RSSI_{smoothed,\,n} = \alpha * RSSI_n + (1-\alpha) * RSSI_{smoothed,\,n-1} + (1-\alpha)^2 * RSSI_{smoothed,\,n-2} + \ldots + (1-\alpha)^m RSSI_{smoothed,\;n-m+1}$$

With $\alpha = \frac{2}{m+1}$ and m is the number of data points used in the smoothing algorithm.

When it comes to selecting which devices to use, we have enough flexibility in our decision for both the beacons (shown in figure 3.12) and the reader. An example of a beacon we could use is the iBeacon from BlueBeam which offers variable TX power options, UID (Unique identifiers) such as Namespace and Instance IDs (for Eddystone UID) or iBeacon UUID (Universally Unique identifier) and Major and Minor IDs, advertising intervals, and has an option that allows us to trigger a broadcast at any time other than its usual advertising cycle. It also allows you to send out the advertising frames under different formats that carry different data depending on the application, such as:

- Eddystone URLs limited to 17 bytes in Eddystone format (protocol specification that defines a Bluetooth low energy (BLE) message format for proximity beacon messages)
- Eddystone TLM packets that can also contain battery information, temperature, number of advertisement frames and time since reboot
- Eddystone UID for broadcasting the ID of the beacon, returning the Namespace and Instance IDs
- iBeacon, Apple's protocol standard returning the iBeacon UUID corresponding to the business that owns the beacon, minor ID which corresponds to the location of the beacon, and major ID which is a more accurate representation of the location of the beacon

Figure 3.12: Bluecharm BLE Beacon with Motion Sensor

For our reader, any device capable of BLE sensing would be enough. This ranges from actual readers, to smartphones, or microcontrollers with Bluetooth functionalities.

3.2.4.2 Sensors

Ultrasonic Sensors: The main advantage of ultrasonic sensors versus other sensors is their ability to detect any object regardless of the nature of the surface. They are also straightforward to integrate with microcontrollers. Ultrasonic sensors would allow a program to specify a distance that would consider an object as being subject to collision. Additionally, the sensors would provide accurate information related to where a user is and how far away they are from a target. For ultrasonic sensors, the most common range of frequency of the ultrasonic pulses spans from 40-70KHz. This frequency determines the range they can cover and accurately detect. Lower frequencies offer a wider range, which spans up to 11m wide with a resolution of 1cm (or lower). For VISION's object detection and user detection around the table, 1cm of resolution would be enough to detect objects that are almost in direct contact with our visually impaired user or to detect the user in the table range. The range would allow us to detect obstructions in the room or a user.

A good example of an ultrasonic sensor (transmitter and receiver) that would fit the design is the HRXL-MaxSonar® - WR™ series shown in figure 3.13. These sensors operate at about 42KHz and can return an output in different forms. The most applicable output of the sensor is a pulse width representation of range with a resolution of 1mm. The range can be extracted using the scale factor of 1uS per mm. It also returns an analog voltage output as a single-ended analog voltage scaled representation of the distance, at a resolution of 5mm or 10mm. The corresponding pin for this output remains at this voltage that directly corresponds to the detected distance. Lastly, it also returns a serial output in an RS232 or TTL format where the distance can read as an integer up to a maximum of 4999mm or 9998mm, depending on the model. Some additional advantages of this series are its low current draw, allowing for a long battery life. Additionally its fast measurement cycles (measurements occur every 50ms on average) are fast enough to detect any new obstacles while the user is advancing towards them. Table 3.2 below summarizes the different models available in this series (“Datasheet for the HRXL-MaxSonar-WR sensor line”).

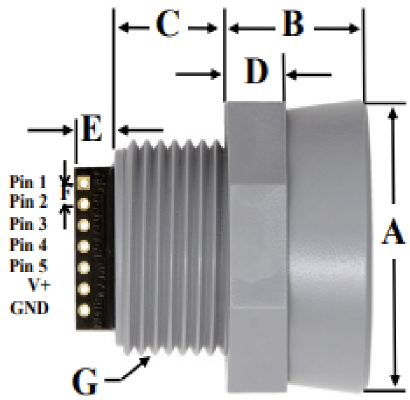
Model Family	Detection Range	Applicatibility
MB7375 and MB7385	30cm to 1.5m	Wider beam from transmitter suitable for closer distances with a broader detection target
MB7360 and MB7380	30cm to 5m	Provides reliable long range detection zones hence used in tank and bin level measurements
MB7363 and MB7383	50cm to 10m	Higher sensitivity hence great to use for applications where objects do not reflect enough ultrasonic sound such as people detection

Table 3.2: Comparison of Different Ultrasonic Sensors

Based on the above table, the third option would be the most suitable option. The main difference between the MB7363 and the MB7383 is that the serial output for the MB7363 is in the RS232 format versus that for MB7383 is in a TTL format. Both RS232 and TTL (transistor-transistor logic) are forms of serial communication where data is transferred between two parties, a receiver and a transmitter, at a specified baud rate, which indicates the speed of said transmission. The MB7383 using TTL serial communication protocol would be the best option due to the following advantages it has over RS232:

- Less susceptible to noise and other interference
- TTL signals’ voltages follow the microcontroller’s voltage supply range of 0 to 3.3/5V whereas RS232 signals are +/- 13V, which would require another external power source
- TTL is hence easier to incorporate with microcontroller designs

RS232’s main advantage is that it allows longer cable lengths to be used, which for obstacle detection would not be necessary considering the sensor would be directly tied to the microcontroller carried by the visually impaired user. RS232 to TTL converters are also readily available in case a switch has to be made between these two serial communication protocols.



A	1.37” dia.	34.7 mm dia.
B	0.70”	17.9 mm
C	0.57”	14.4 mm
D	0.31”	7.9 mm
E	0.23”	5.8 mm
F	0.1”	2.54 mm
G	3/4”-14 NPS	
H	1.032” dia.	26.2 mm dia.
I	1.37”	34.8 mm
Weight, 1.23 oz., 32 grams		

Values Are Nominal

Figure 3.13: Model and Dimensions of Compact Housing HRXL-MaxSonar Model (Permission Received from Maxbotix)

<u>*IR Sensors:*</u> Compared to ultrasonic sensors which all rely on the time-of-flight principle, other IR sensors use different mechanisms for their functionality. One of which is triangulation. Infrared LED triangulation sensors determine the position and distance from the object using geometric considerations. A collimated laser source (transmitter) is used to illuminate the object to be measured. The light is reflected back (receiver) and focused by a position sensitive detector (PSD) comprising small photo sensors in a row called pixels. The distance is then measured using a ratio of the product of the distances over the size of the detection pixel. The main issue with this approach is its reliance on a different factors lowering its resolution at larger distances. Its biggest perk being the lowest prices comparatively for sensors.

Time-of-flight IR sensors, on the other hand, similar to ultrasonic sensors, operate by sending a light pulse to the object and determine its distance based on the time it took to reach the detector. They have a much longer range than their triangulation counterparts, along with other benefits such as faster transmission and reception times, rapid refresh rate, and lower power consumption. The main disadvantage here is the increase in price and the inability to differentiate targets. It cannot be used for object detection, but would be a great choice for collision detection.

A good option that would fulfill the above advantages without a huge increase in price is the VL53L0X (shown in figure 3.14) from STMicroelectronics whose range of detection goes from 50mm to 1200mm (or 2000mm in one of its function modes) (“World's smallest Time-of-Flight ranging and gesture detection sensor”) which is more than enough for collision detection. Its 940 nm VCSEL emitter (Vertical Cavity Surface-Emitting Laser), is invisible to the human eye. Coupled with internal physical infrared filters offering higher immunity to ambient light, and better robustness to cover glass optical crosstalk. The output can be obtained either using a polling or interrupt mechanism, allowing it to be programmed to consistently check for any obstacles on the user’s path and sending the analog output through I2C communication to the main microcontroller in use for the design. The default timing for initialization, measurement/ranging and other housekeeping functions it performs is about 33ms, which offers more than enough time to detect any object in the user's vicinity as he navigates towards the pool table. It also uses a streamlined beam that would make detecting a user positioned directly in front of the time of flight sensor much easier. This feature is actually a main disadvantage when it comes down to detecting obstacles. Being a laser-based system, the transmitter sends out a straight line laser and only detects objects in the very narrow beam (25 degrees Field Of View). Positioning of the sensor would be of great importance when trying to maximize obstacle detection (this is investigated in a later section). Another noteworthy advantage is the low power consumption of about 5-6 μA in standby mode. There are a few other considerations such as the nature of the material and the color of the material which affects the accuracy of the measurements. These factors will be taken into consideration when designing the system and tested to determine how much this affects detecting objects or people.

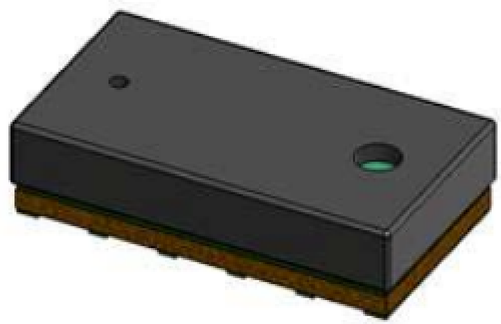


Figure 3.14: VL53L0X Time-of-Flight Ranging and Gesture Detection Sensor (Awaiting Permission from Digi-Key)

Based on the last 2 sections, a combination of both ultrasonic and IR sensors would be the best alternative for object detection/collision avoidance. Lighting conditions affect IR sensors while ultrasonic sensors are not affected by this. Ultrasonic sensors are reliant on the shape of the target, struggling with soft, curved, or thin objects while IR sensors work fine under these conditions. Ultrasonic sensors are not easily able to detect sound absorbing surfaces such as clothes or other fabrics hence would struggle to detect human presence in non-ideal circumstances.

Other Sensor Technologies: There are more available options for sensor technologies that we considered, but that would not fit the scheme of the project. Here we discuss a few honorable mentions and reasons why those avenues were not pursued for our physical design before making a final decision on which technology would be used for user and/or obstacle detection.

Conventional or linear ultrasonic sensors only record one-dimensional data, usually the distance from the sensor to the detected object. Other coordinates such as inclination or elevation angles of objects are not calculated with by these sensors, leading to potential inaccuracies in detection objects such as the curb and low-lying obstacles depending on the location of our 1-D sensors. 3-D ultrasonic sensors, on the other hand provide 3-D coordinates for objects reflecting the ultrasonic pulse, hence providing horizontal and vertical coordinates for where the object is located and detecting in a broader field of vision. A lot of the same concepts explained earlier for linear ultrasonic sensors would also apply in this case, including the advantages and disadvantages of this.

Some of the most advanced 3-D ultrasonic sensors, such as the ones developed by Toposens, use a concept similar to echolocation in bats. (Nancy Seckel) The sound transmitter sends out an ultrasonic pulse. The pulse is reflected by surrounding objects and received by a sound receiver. The pulse received is then interpreted by a sound processing unit and 3d coordinates are calculated based on the time it took for the echoes to arrive back to the microphone/unit. This gives us a total range of about 160 degrees, which would be enough to cover half of a table side at a single time. Two of them covering both sides of the table and detecting the user at any point, or one of them mounted on a rotating frame would be enough to detect everyone around the table in a quicker manner than it would take using a linear beamed 1-D ultrasonic sensor. Other interesting features of the Toposens model are that it allows detection of up to 3000 mm or 300 cm with a resolution range of 1 cm - 5 cm, has a field of view of ±80 ° Horizontal and ±40 ° Vertical. However, these major advantages are not a good enough justification to oversee the increase in price and complexity of using this versus any other sensor. Also, depending on the physical design discussed later in this document, we can cover the blind zones incurred by using 1-D sensors, which is the main advantage of using 3-D sensors instead.

LiDAR makes use of echo-reflection using laser beams in the near infrared, ultraviolet or visible spectrum. A LiDAR system measures the time it takes for emitted light to travel to the ground and back. That time is used to calculate distance traveled. One of the major advantages of LiDAR compared to other sensor types described earlier is that LiDAR does not depend on lighting conditions and a much broader detection range that will work great for user detection in a room, but would be irrelevant when it comes to obstacle detection or navigation around the table. Most of the available options are however, much more expensive for the better quality ones and are much more scarce than either ultrasonic or IR sensors.

A fairly in-budget example that is worth mentioning is the SmartLam TP-Solar 0.2-12M LiDAR Sensor. It offers a detection range going from 0.2m-12m with a blind zone for any object below the 0.2m range and a range accuracy of 5cm and above and a resolution of 1mm. In this blind zone, all distances would be invalid. This sensor only offers a field of vision of about 1 to 2 degrees which is much lower than the aforementioned 160 degrees field of vision for 3-D ultrasonic sensors and even much lower than the field of vision of 1D ultrasonic sensors that are in the tens of degrees. Other LiDAR systems may have a much better field of vision in comparison, but it once more comes down to the a much higher price without any considerable reasoning behind the price gap for our applications. Going through a few other available LiDAR sensors, even doubling or tripling the price as compared to the example referred to in this paragraph only gives us about 3 degrees for our field of vision.

Spinning LiDARs would allow us to detect all obstacles in a near 360 degrees range, with an angular resolution of around 1 or 2 degrees for most sensors in the market. Also, they are prominently used in the market for object detection already, as seen in automotive vehicles. However, once again, the cost does not justify using LiDARs for our application.

RADAR works in similar fashion to ultrasonic navigation, replacing the acoustic waves with radio waves and estimating distance using as the signals bounce back from nearby obstacles. In this kind of receivers, the impulse acoustic wave response of the receiver is the same shape and the receiver essentially measures the degree to which the received signal and the transmitted signals are correlated with each other. RADAR systems are, however, much slower, more expensive and more complex than ultrasonic systems.

3.2.4.3 Localization Algorithms

Sensory input is fundamental for user localization within this project, but the proper algorithms and computational methodology to support the inputted sensory data is key to having accurate data to transmit for proper guidance commands to be sent to the user. Inputs resulting from each sensor type all have the goal of understanding where the user is relative to the billiards table as a whole. To do this, several back end processes can be explored to achieve the desired goal of visualizing the table environment and localizing the user with respect to common data points.

SLAM: In the field of autonomous navigation of robots and automobiles, simultaneous localization and mapping (SLAM) is an improving asset for real time responses to a system's surroundings. SLAM works with sensory imagery primarily from cameras or LiDAR to be able to map the present area and, in the same instance, localize the system relative to the area it navigates through. This goal is best realized through path finding algorithms and object avoidance (discussed further in Section 3.2.5 - User Guidance), making it a great asset for real time responses of autonomous vehicles for terrain that can not be previously predicted ("What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink - MATLAB & Simulink").

Maze Array: To do this, constant variables must be set based on the type of interface that is inputting data to this processor. Constants of interest are the size of the table and position of origin point of the sensors and the variable of interest is the changing distance determined between the sensor(s) and the user. With these variables, an accurate localized position in a two dimensional space can be achieved and easily exported with limited size of data being transferred.

In the case of an array being propagated for localization and path guidance of the user, an important distinction to be made lies with the choice on how large each array position is, how accurately to portray the user within these positions, and how many positions deep to make the array. A diagram of such a representation is shown in figure 3.15, where a graphical interface housing the current layout of the billiards table and its accurate physical space would be outlined by a two dimensional array housing the location of a user. Relating to the constraints of such a model and why the variables described house trade offs comes from the desire for accurate real time updates of such an array for both the display and guidance system. Simplistic approaches housing vast approximations for location will be simple to calculate and communicate but risk giving an inaccurate representation that may hinder a user from proper navigation. On the other hand, a very in depth set of data points will add more complexity to the data that is communicated. At such low levels of data communication, lag in communication is not a grave concern and can be considered as lower priority. Specification and ideal frequency of updates to the proper load times is of a higher concern when it comes to efficiency, which is a task that can be optimized within embedded controls.

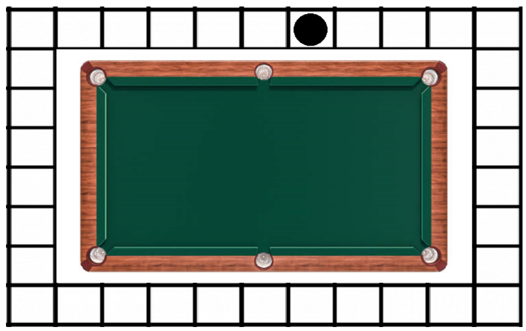


Figure 3.15: Localization Algorithm Array Scheme

More complex localization approaches can also be considered. A three-dimensional space adds significantly greater hurdles to the amount of data that must be communicated, the number of sensors that must be present, and the communication speed of the data. Given the nature of the guidance system and the desire for speed over complex representation, a method such as this may not be optimal for the constraints of this project. However, an added benefit of a three-dimensional state space comes in the feature of object avoidance and recognition. If an outside variable such as an added person/guide enters the frame of sensory inputs, it may be hard to localize the desired user compared to the confounding inputs.

3.2.5 User Guidance

Corresponding with user localization is the outputs to navigate the user to the desired location of the next shot on the table. Previous assistive technology has deployed navigation methods that can be augmented to our desired specification and constraints. Similarly to the approach for user localization, guidance methodologies carry various pros and cons that can be weighed by comparable variables of cost, scale, accuracy, ease on the user, computational bandwidth, and corresponding algorithms. To explore possible routes for this technology, previous technologies in audio and sensational guidance have been explored and come in varying extents and approaches.

Summary of Requirements:

- System can position user within 1 foot of the desired location
- User is oriented within 15 degrees of the desired shooting direction

3.2.5.1 Audio Outputs

One of the most intuitive guidance systems for user guidance for the visually impaired centers on audio outputs. As mentioned in several previous projects discussed in the visual impairment assistive technology section, voice commands are a very common method of guidance in a real world setting where many unpredictable variables may occur. Alternatively, for the case of navigating a stationary table, simplified methods may be deployed. For instance, audio that is outputted merely to navigate a user by a constant sound in the direction of the destination can house value, and an altering pitch tone could help differentiate the concept of distance from the destination to the user. While these simplistic approaches can seem intuitive to an individual with knowledge of the make of the system, a new user may not comprehend elementary instructions being presented as easily. Applications such as this may require some form of preliminary explanation to the user of how the system operates, while more complicated approaches such as audio commands would in fact be intuitive to the user. For example, an output stating “turn right”, “move forward”, or “stop” are messages easily understood, while a constant tone on the other hand has no intrinsic meaning.

Command-Based Audio Output: Factoring into these audio approaches is the delivery method and density of said method within the system. For an instruction based output, the sources of the output do not necessarily have to be distributed. A centralized location either on the user or in a constant position that emits the instructions is sufficient. However, benefits based on the orientation of the user may arise in having a centralized output of instructions to not confuse delivered instructions. Inconveniences can arise in cases where a central location is emitting sound from a position that is opposite of the direction the instruction is oriented towards. The severity of a case like this is minor in the presence of a robust algorithm that will continue to guide the user based on their adjusting location. A design such as this could also reflect closely with home voice assistant devices such as the Amazon Alexa and Google Home Mini. These devices are recommended to be placed at a central location in the house both for recognizing audio commands and for proper delivery of corresponding outputs. A system such as this realizes two way communication and holds value in terms of the potential to introduce audio commands on top of audio guidance.

A user centered approach as discussed in the previous visual impaired assistive technology section (“Guidance System for Visually Impaired People”) discusses the use of headphones for communicating commands to the user. A user based approach can be easily deployed with the latest wireless technology within a bluetooth headset. Commands can be communicated from a central processor located outside a user and sent via bluetooth. This decentralized approach to command-based audio eliminates the factor of distractions brought by centralized audio.

In addition to command outputs, the described systems can also be relevant in the realm of relaying outcome information. For instance, in the case of a user conducting a shot, having additional audio that confirms the resulting success or failure could have value to a user that cannot see or visually comprehend what has occurred. This is similar to how previous projects have utilized gTTS ("Guidance System for Visually Impaired People") API for command based navigation or the use of the same API for outputting the words of a written page ("Reading Device for Blind People using Python, OCR and GTTS"), but the same practice can be extrapolated for any situation. As the number of outputted results has a finite value, this feature can hold value for a user in the command-based model of output as it requires identical materials as need to be present for this system.

Direction-Based Audio Output: In the case of a simplistic audio approach for directional commands, a distributed network of speakers could be deployed across the realm of navigation for a user. This array can be deployed in various manners depending on desired accuracy. In the case of navigating a table, the baseline requirements would settle upon the four corners of the table having speakers to be able to deliver a command for each 2-D direction around the space. This can be made more accurate if added speakers are added between corners of the table to better position the user at a desired location. Additionally, the accuracy can be enhanced in the alternative manner of having the speakers emit varying levels of pitch to describe distances. For instance, higher pitch could mean further distance to travel and lower pitch could relate to approaching the desired location. These varying implementations also come with a tradeoff in cost based on a linear increase with the added number of speakers in the array or the cost increase from added complexity of the audio technology.

To illustrate the discussed audio delivery methods, figure 3.16 showcases the hypothetical case of a user attempting to navigate from the upper left-hand corner to a desired location of the table. The three audio output mechanisms are shown within the graphic with corresponding labels and expected commands based on their varying purposes. The array-based output is implemented at the basecase of four corner speakers.

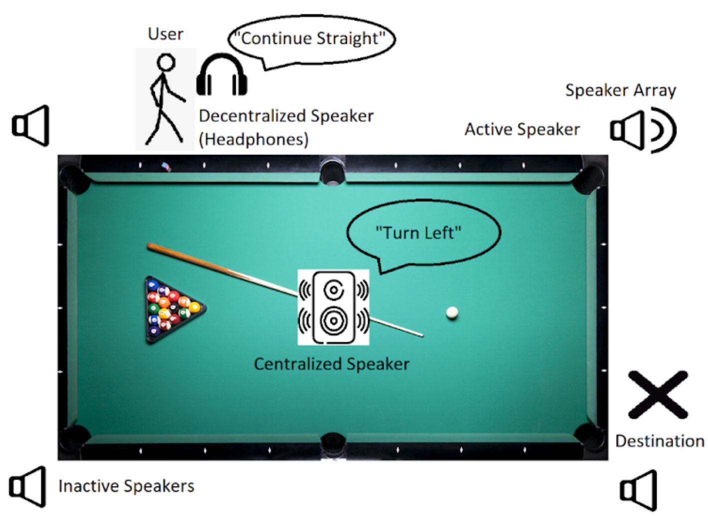


Figure 3.16: Audio Based Navigation Mechanisms

Audio Aim Guidance: Once the user is guided to the proper position on the table, they must then be oriented toward the ball. This mechanism can be deployed in similar approaches as the positional guidance discussed.Within a command based mechanism, real time orientation data is a necessity as corrections to the left or right of the user can only be comprehended if a feedback of data is present. The audio array method comes with the limitation of the same degree if deployed at the base case of four corner speakers. Corrections will also be challenging in this case due to both the wide spacing of the speakers and the algorithmic control of which to activate based on the varying possible positions. To improve accuracy of an array for aiming the user's shot orientation, a denser population of speakers is a simple enhancement. At the worst case, the possible blind spot for shooting position is rather wide, and will lead to challenges with the hand off to the user side apparatus of SCRATCH. To limit this challenge and ease difficulty on the user, a worst case angular error from the desired shot position should be established and then used to determine the necessary density of speakers.

Audio Levels: If audio is used for guidance of visually impaired individuals, audio levels produced should be considered for both the ease of proper distinguishment of commands and for auditory wellbeing and safety of the user. Audio levels should be adjusted after installation within multiple environmental settings to confirm they meet these specifications for the user. Some systems can even be implemented that utilize feedback loops for gain control of outputs with installed microphones. (*Accessible Pedestrian Signals #*) For the case of VISION, this specification does not need to be considered down to a predetermined decibel level, but instead needs to be standardized across all the speakers and adjusted within the validation process of the project.

3.2.5.2 Physical Sensory Outputs

While audio has been explored as a guidance mechanism for users with limited use of their site, an additional sense can be deployed in the sensational awareness of a user's surroundings. Stemming from the use of probing canes for the blind, the technology of physical feedback to visually impaired individuals has grown a great deal with the improvement of technology. Vibrations can now be actively created utilizing haptics to deliver purposeful information to a user that describes actions to take or a direction to move.

Designs like that of Maptic ("Maptic is a wearable navigation system for visually impaired people") shown in figure 3.17 have been deployed in wider variable environments for guidance in everyday tasks. This technology is worn by the user in what appears to be simple accessories but instead is a useful haptic guide for the visually impaired. Optical sensors within a necklace-worn device take in inputs that are then routed through an iOS application that sends signals to each of the wrist feedback devices. These signals can be configured in various manners to transmit information and can also be interfaced through voice control. Systems of this manner are very beneficial for guidance in a changing environment such as the open world, and can be extrapolated for more defined scopes.

Figure 3.17: Maptic Haptic Feedback Apparatus ("Maptic is a wearable navigation system for visually impaired people") (Awaiting Permission from Dezeen)

Within a different scope of problems for the visually impaired, the University of Maryland conducted research into a project giving the blind better ability to parse through reading text off a page shown in figure 3.18. Haptic feedback was used in the study as a manner to deliver information on the page layout and used a camera to take in the text information on the page. ("Evaluating Haptic and Auditory Directional Guidance to Assist Blind People in Reading Printed Text Using Finger-Mounted Cameras") This technology approaches haptics from a different direction, but does show how minimal information transfer from vibrations can be used in conjunction with additional technologies to achieve enhancements in the lives of the handicapped, similar to the goal of VISION.

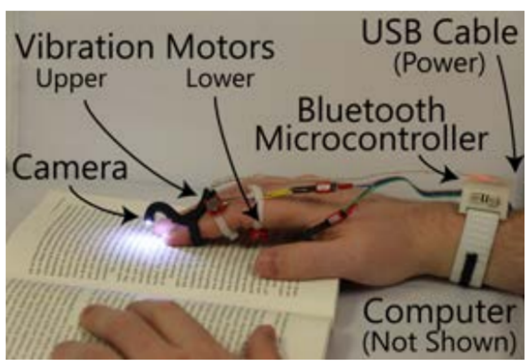


Figure 3.18: Hindsight Haptic Feedback Apparatus ("Evaluating Haptic and Auditory Directional Guidance to Assist Blind People in Reading Printed Text Using Finger-Mounted Cameras") (Permission Graned from Lee Stearns)

Within the scope of guidance to desired shot locations on the pool table, commands can be delivered to the user that mean move left, right, forward, backward. As there is no locational specific information being delivered however, this can present comprehension hurdles. A new user may very well misunderstand a command being delivered and struggle to easily follow commands. Additionally, angular orientation of the user creates the need for haptics to require a sort of correction based on this parameter for proper positional guidance. With this variety of commands being delivered in a base level that is binary at the simplest level and can be enhanced with more feedback devices, it can be seen that design can quickly divulge into complication and result in a negative user experience. These factors must be considered in design, especially when weighing options in a static vs dynamically changing environment.

3.2.5.3 Guidance Algorithms

Navigation algorithms that bridge the gap between sensors to output is the glue to a complete navigation system for an impaired user. Algorithmic constraints are examined with the assumption that an accurate user location and the desired location is being polled to the guidance system from the user localized functionality of the system and the billiards AI respectively. The goal of the guidance algorithm will be to locate the shortest path between these two data points and navigate around main item obstacles such as the billiards table. Obstacle avoidance is a viable feature to explore, but may create significant added complexity to tools deployed for user localization. This being the case, this feature is considered a stretch goal of the project. Once the desired path is determined from source to destination, outputs must be accurately relayed to the user based on the delivery mechanism for user guidance.

<u>*2-D Space Traversal:*</u> To navigate the table safely, a leading mechanism to realize the system space is a two-dimensional created similarly to a rudimentary maze that outlines the table as a boundary the user cannot navigate through. This can be accomplished by utilizing common algorithms for navigating a 2-D matrix. There are several approaches to realize this goal including including the commonly deployed backtracking “Rat in a Maze” algorithm. The simplest form of this algorithm will continue to test paths in a binary maze where 0 is traversable and 1 is an obstacle until it reaches the desired location. As higher processing power and a shortest path is desired for this test case, an algorithm of this sort will want to find the absolute shortest path between two points and will want to terminate the function as this path is determined to not hinder processing ability. The Rat in a Maze algorithm operates at $O(2^{n^2})$, meaning that a large array will lead to a nontrivial run-time and severely hinder computational speed (“Rat in a Maze | Backtracking-2”).

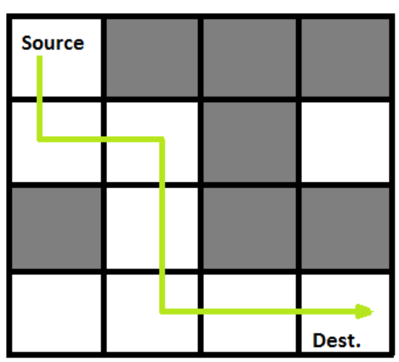


Figure 3.19 Maze Traversal Example

While maze traversal can be a very useful method for complex and changing 2D arrays, the specific use case of VISION brings up the option for an alternative method. As there is a static grid in place that is centered around a constant dimensional table, there are only two available paths that can be taken to navigate the table's perimeter at any given time. With this being the case, a binary guidance algorithm can be deployed, which flows in one of two directions. This approach removes the need for complex computational calculations and puts the strain of the system on sensory input processing.

To deploy the above algorithms, a 2-D space must be accurately created prior to the start of navigation. For this to occur, a constant center point should be established relative to the user. This point can be located at any spot, but must be adjusted accordingly if to have an accurate location of a moving user. The proper state of value of each square of the maze must be set. Recognizing the constants that will not change in this system centers on the billiards table and any added obstacles that may be present within the space. By noting these, the requirement to sense the location of the table is relinquished from the system. Determining the constants would depend largely on added design of the system and dimensions of the table. In addition to these determinations, a determination should be made on the size of each array value. This can be relative to the size of the average human, and can be larger or smaller depending on the expected accuracy of sensors and the desired accuracy of positioning the user.

Obstacle Avoidance: If an unexpected object is discovered to be on the floor around the table, warnings and alternative paths can be deployed. The primary limiting factor to this approach is certain deployed sensors will be either robust to these obstacles or their localization algorithms will be greatly hindered. To definitively differentiate between a user and an obstacle, a mixed sensor approach as described in the localization algorithm section would ideally be deployed. In the case where an obstacle is localized, this factor can be added to the 2-D space as a present array value and algorithms can be deployed to avoid its presence. A system like this can be complex if it requires stepping into a dimension outside of direct adjacency to the table, and would not be compatible with simple guidance mechanisms.

3.2.6 Feedback System

The feedback system will be based on sound in order to accommodate the vision impaired player. The table should give the user feedback on the following events: If a game ball is made, a scratch, or if the game is won. The table will feature a speaker at every pocket, this will allow the player to be able to determine which pocket the ball went into. The following research is to find ways in which we can implement such a system.

Event Sensing: This is the process of discovering if a shot was made by the user. It must also be able to determine if the game has finished or if there was a scratch on the user's turn. There are two main ways in which we would be able to determine if an event has occurred, one is through our computer vision system while the other would be setting up sensors in every pocket.

By employing our computer vision system based on the research in section 3.2 of this paper, we will be able to use that information in order to alert the player when an event occurs. This will prove to be higher latency than an approach using physical sensors on every pocket. However, the computer vision algorithm will have to be improved to meet extra requirements. The first requirement is that it must be able to communicate that a ball has been pocketed. It must also allow for detection of a scratch, this means the computer vision system must be able to distinguish the cue ball from the normal ball. Despite these drawbacks, employing the computer vision system to assist in result feedback would offer a major cost advantage, as well as a possible development time advantage.
A sensor based system would allow for almost immediate feedback to the user. The sensor would have to be present within the pocket and be able to withstand a hit from the pool balls. That is not ideal as the sensors will likely be fragile. Some possible options for sensors are a force sensitive resistor (FSR) and an RFID tag.

The FSR would be a good way to detect changes in pressure when the ball falls into the pocket. The FSR works as a variable resistor and an example is shown in Figure 3.20. It has virtually infinite resistance when not presse. As it is pressed with more force however, the resistance quickly goes down. The FSR has a conductive polymer that allows for the change in resistance when a force is applied. This approach is however not feasible unless the pocketed ball was taken out after the shot has been made. Otherwise the system would have no way of knowing whether or not another shot has been made in the same pocket. One way around this inconvenience would be to keep track of the current value, if it goes up the proper amount for another ball being made, then you could give the user feedback once again. However this will be difficult, as the function for force compared to resistance is not linear.

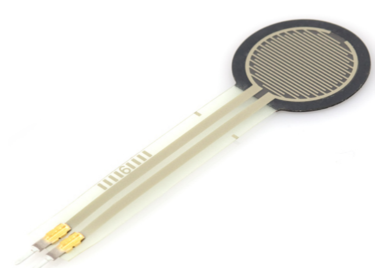


Figure 3.20: Force Resistive Sensor (Awaiting Permission from SparkFun)

As discussed in the RFID section, these chips could be placed inside of the ball for the purpose of detecting if a ball were to fall into a pocket. The range requirements would have to be met in a way to ensure that a ball very close to a pocket would not prematurely be counted as a made shot. The other downside of this is that it would likely require a very tedious process to place the RFID tag inside of the pool balls. Doing this without disrupting the natural movement of the balls after the modifications would also require extreme care. A solution using this approach can be found when examining how golf driving ranges are able to track many metrics on a user's shot. By using RFID technology, the user can see the speed of their ball, the path, and the top height traveled by the ball. One such company known as "Top Golf" employs Impinj M700 Series RAIN RFID tag and is shown in figure 3.21. The technology they use is proprietary, however, we know that each ball has a RFID chip that is programmed before the shot is taken, along with a series of sensors in the field in order to gather the metrics previously described. An approach similar to the one taken by Top Golf would be very valuable. However, the room for error on a driving range is many yards, while the room for error on a pool table could be a centimeter. Currently a patent has been granted for using RFID technology to create a score tracking system for the game of pool, but without any commercial offerings or viable demonstrations on the effectiveness of this technology for pool, we are unable to see the true effectiveness of this approach.

Figure 3.21 RFID Tag Embedded in Golf Ball (Awaiting Permission from Ok-Chemistry-2194)

<u>*Feedback Sound System:*</u> The sound system will consist of a speaker located at each pocket. This is to allow the player to orient themselves to the pocket which the ball has fallen into. Some requirements for the sound system are: volume level sufficient to distinguish pocket location from approximately 13 feet away (9 foot pool table with included 4 foot buffer) and six speakers, one at each pocket.

The feedback system must also handle the case in which more than one ball is made. If two or more shots are made into a pocket, the shots will be placed into a queue and announced in sequential order. In the case that an eight ball is pocketed, the system will end the game before further shots will be announced. The edge case in this scenario will be if two balls enter the same pocket, this may be difficult to distinguish based on the range of the RFID technology used. If the technology is capable of detecting two balls in the same pocket, then this case will follow the same queue system. A chart showing the progression of events is shown in figure 3.22.

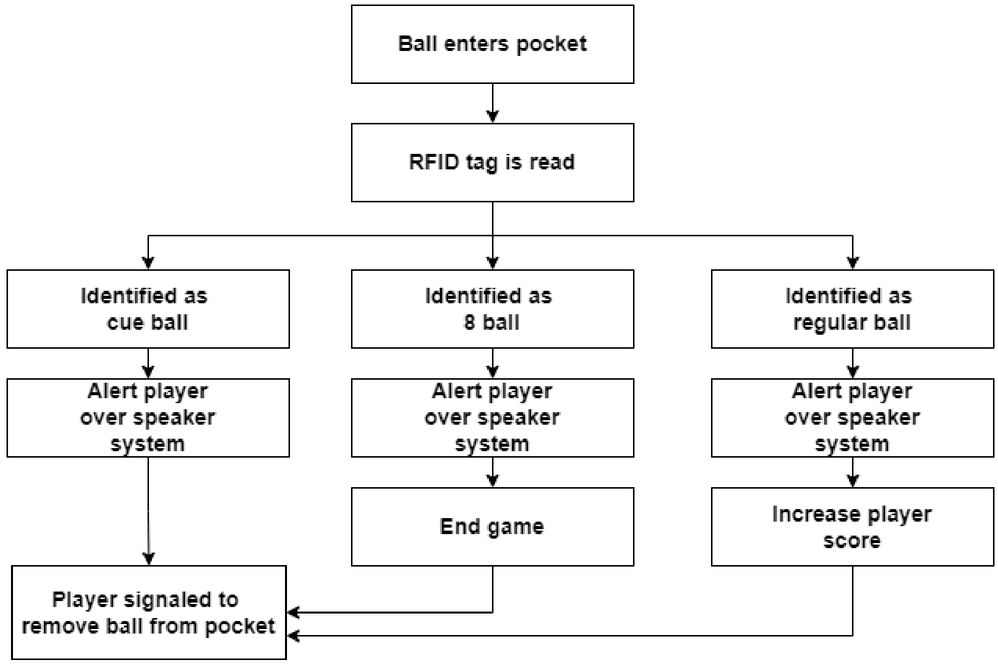


Figure 3.22: Feedback System Shot Results

Determining Shot Results: The feedback system needs to determine what occurred during the player's previous shot attempt. The possible shot outcomes are shown in figure 3.22. In order to determine if balls were sunk during the previous shot, the feedback system will compare the current state of the billiard table to the previous state of the billiard table. The table state information will come from the output CSV files generated from the computer vision system. These files will contain all the information necessary to determine what billiard balls are no longer present after the player's turn.

The system will parse each file and determine if the cue ball is present, if the eight ball is present, how many green balls are present, and how many blue balls are present. Comparing the previous table state date to the current table state data will determine which of the five possible scenarios the player's shot falls under. The results of this comparison will determine if the player must continue playing, has won, or has lost. The logic used for this comparison depends on if the eight ball is present. If the eight ball is not present, the user wins if they have no more game balls or loses if they have one or more game balls. If the eight ball is still present, the user continues playing and is notified if they did not make a ball, make their game ball, or make the opponent's game ball.

3.2.7 Direct User Commands

Within this system, the goal is for the user to have as many assets as can be provided for giving them safe and clear access to be able to navigate the pool table and have an understanding of where they are at all times. In addition to system side navigation and localization techniques, commands sent by the user and/or a secondary controller can be explored and implemented when the most benefit to the player can be realized. These commands can be implemented either for scope critical actions such as designating the end of a turn or focused on enhancing the user experience. For this purpose, previously deployed technology in remote controllers, centralized control, and audio commands are researched for viability within this system. The possible benefits of these designs will have their importance weighed for our system for an optimized user experience.

3.2.7.1 Control Interfaces

The scope of VISION encompasses certain baseline commands that will require user interaction. Whether these commands are relayed from an assistant or the user directly, they will be critical to the performance of the system.

Remote Controller: A possible additional asset for the user within this scope comes in the deployment of a device that stays attached to the user that primarily can be used for setting basic commands of the system. This type of remote controller could also have the added benefit of being accompanied on the same devices that define user localization techniques previously described in device to device communication methods. Controllers located on a user have been referenced in the section discussing visual impairment assistive technology and additionally correlates to the concept of remotes used for items such as navigating a television interface with touch integrated controls. The latter can be of importance in basic design of remote interfaces for the reason of allowing visually impaired users the ability to at a minimum have an understanding of and be able to control critical functionality of a system ("Ensure that the remote control can be used without requiring sight"). Remotes such as the one shown in figure 3.23 showcase how a basic interface for control over an audio interface could be made intuitive for a blind user with limited guidance. The simple setup with raised and shaped buttons has been used to relay the intent of controls to users at scale for many years. This importance can be mirrored relative to this system for the needs of critical tasks that a user may need at any point in their performance. A similar design could be extrapolated to use within VISION with proper distinctions of commands in place. For the optimal user experience, having intuitive control directly from the source user allows for the quickest response and a superior experience.

Figure 3.23: TV Remote or the Visually Impaired ("Tek Pal Tactile Low Vision TV Remote Control") (Awaiting Permission from Maxi-Aids)

Centralized Control: In contrast to an interface local to the user, centralized control would require a non-impaired assistant to be in place and be able to relay commands for the current process in place. A centralized interface could be located either on the table, on the side of it, or distanced from the table. This interface would have a focus on buttons or other methods of communicating intent to the primary processor. This could contain critical commands, audio preferences, display settings, etc. While the remote controller is possibly a more optimal method for late stage development of products, a centralized control interface could be a better fit for a prototype to determine where limitations on commands may be. Additionally, having an assistant is most likely a necessity for early stage testing, which would eliminate the benefit brought on by a fully user side interface.

3.2.7.2 Audio Commands

One of the more common features of previously deployed blind-assist technology was the ability for users to communicate their desired system task via voice commands. Previous technology in this field utilized Python speech recognition packages to allow for user commands to be read in and interpreted by a processor and respond accordingly ("Guidance System for Visually Impaired People"). An interface such as this is an advanced feature that has benefits and distractions. The most outstanding benefit of this interface is the ease in being able to ask questions and send commands that is more intuitive than feeling for a proper command on a user side remote and attempting to understand the intent of each button. Additionally, a proper audio command interface would possibly have the ability to interpret approximate ideas from inaccurate commands and comprehend a best course of action. While these factors of ease are valuable, factors of noise pollution both from surrounding environments and from deployed audio guidance methods introduce potent constraints and problems to the system. Issues of this manner can be addressed with proper filtering and close proximity mics, but is a rather expansive problem to combat.

Commands of Interest: Determining the most crucial commands for the use case of the augmented billiards game being deployed in this system requires a weighing of the trade off between the simplicity of the interface and necessity of each command. There is a wide spectrum of possible commands that can be of use to a user and assistant. At a baseline, there are commands required for basic functionality of the game to occur, and others that are more centered on aiding the user experience. Some possible commands to explore include: Center User, Start Game, Shot Taken, Game Status, Begin Navigation, Pause Game, Reset Game. These commands could correspond with responses from a centralized speaker system, begin a guidance system, or allow for a reset process to commence.

3.2.8 Visual Display

The display allows for users to quickly look at the angle and force level required for the best possible shot selection from their current position. It will also allow for the VISION team to debug in real time to verify that the computer vision system is mapping locations correctly as well as verify the shot selection algorithm is coming up with reasonable outputs. This system relies on the computer vision system to give it the locations of each ball and the shot selection algorithm to give the display system an angle and force.

Monitor: One of the most practical ways to display the output is to use a computer monitor. Monitors are relatively inexpensive and offer high-quality displays. The Jetson Nano natively supports USB 3.0 and HDMI interfaces. Nearly all modern monitors support HDMI, so finding a compatible monitor should not be an issue. Another benefit to using a monitor as a display is that the display does not need to be located near the Jetson Nano. HDMI supports cord lengths of up to fifty feet without any major signal loss (Herrman). If HDMI is used, this will allow for the monitor to be located away from the billiards table so that it does not become an obstacle to the player. The ideal monitor will support HDMI, a refresh rate of at least 60Hz, and be at least 20 inches.

Graphical User Interface (GUI): The graphics can either be produced locally on our computer system or by accessing a website. Some of the options for this system are below. The system used also does not have to be high performance as we will not be doing any physics simulation or gameplay mechanics. The system will only take in the location of the pool balls and map them as well as show the next shot.

Desktop Application Development: Desktop applications refer to applications run from executables on a local computer system.

Pygame is an open source set of Python modules which allows for game development. Python is not the optimal choice for creating GUIs as the code is much slower than alternatives such as Java and C++, but it has many modules written in C in order to speed up runtime. Pygame was used by many of the developers to create the pool simulators discussed in section 3.2.1 labeled “simulation tools”. Pygame is mainly for 2D development of simple games, making it a strong candidate to represent our table. We would be able to take the existing open source simulation projects and remove the excess functionality.

JavaFX is an open source project for developing GUIs in Java. Has an interactive design system called Scene Builder for designing the look of the interface and allows for unit testing through another program called TestFX. There is also a separate game engine for JavaFX called FXGL, but the base JavaFX will most likely meet our needs.

Unity is a framework that allows for 2D and 3D game development. A free version is available for personal projects and has a vast amount of assets that may be easily integrated. Unity will allow for quick development of visuals and has a large amount of community made graphics. The high performance and optimization of Unity will give a strong advantage in terms of usability. Scripts are written in C# and allow for almost all functionality that traditional c# programs would offer.

Web Application Development: A web application is one that is hosted on a server and requested by a client via the internet. It offers an easier way to communicate wirelessly.

React.js is a javascript based library for creating user interfaces. It allows for you to create components which can easily be reused for quick development. React is a very common library used in modern web development and is featured in many popular web development stacks such as MERN (MongoDB, Express, React, Node). However for a simple one page application, using React may not be the best approach. However React is made for interactive and quickly changing UIs soon as data changes. A common plus side for using React is that you will use javascript for development. Having the same language for both front and backend development allows for faster development.

Javascript, CSS, and HTML may be the best option for such a simple GUI. In order to create tables we could use simple CSS shapes to create the table and pool balls. The pool balls could have their location on the screen programmatically changed whenever a new event occurs. One event would be the CV system sending the data for the location of every pool ball, the other event would be once an optimal shot was selected. The display would then refresh and show the locations of the ball and shot selection asynchronously.

<u>GUI programming for embedded systems:</u> If our design favors programming our GUI onto a microcontroller instead of a microprocessor we will have to use a much different approach. We will not have the same breadth of applications to choose from for developing the GUI, but the cost will be significantly less.

Qt is a GUI development software written for many different applications including mobile and desktop. This particular version is geared towards a high performance GUI for microcontrollers. The downside of Qt is that only four hardware platforms are currently compatible with the software across 10 different devices. All of the MCUs are higher end performance and therefore increased cost. While costs may still be less than a microprocessor, it will not be as large a gap as for a generic microcontroller.

<u>Communication between systems</u>**:** The communication between the computer vision system and the shot selection algorithm can be done either through wireless or wired communication. Wired communication will limit the GUI to a desktop application for simplicity, while the wireless option will allow for easy development of both desktop and web applications.

<u>Wired communication</u>**:** The use of a communication protocol such as SPI, USB, or I2C may be used in order to communicate directly with the display systems computer. The exact communication protocol is mostly subject to our overall system design and does not have a significant impact for the display system. We will instead be researching microcontrollers which have the ability to display a functioning GUI.

The top candidate for the processor is a Jetson Nano. The Jetson nano and all of its specific requirements have been covered in section 3.2.11. The nano has a built-in port for HDMI, meaning we may directly plug it into our electronic display. This enables the ability to run a multi-threaded application, both the GUI and the computer vision system must run asynchronously on the same system. The Jetson has 128 cores, therefore it is likely that it will be able to accommodate the running of two applications at once.

The operating system for the Jetson is based on Ubuntu Linux, meaning that it will be compatible with many of our GUI applications. The Jetson can be run in headless operation mode, which would turn off the display, or in normal operation, with the Jetson displaying its own GUI. Another great feature of the Jetson is its ability to use the internet to broadcast our information to a website. This would allow for the website dashboard to be seen from a mobile device as well as any computer connected to an internet network.

3.2.9 Absolute Orientation

For means of getting the most accurate shot direction orientation, designating a position and direction that are defined absolute relative to a given point will allow for the most accurate dissemination for user side system commands. Following the general directional guidance of the user to the proper location, orientation relative to that point is crucial to the user's ability to have a chance at properly hitting the cue ball. To get metrics required to relay this information both to the table and user guidance systems, establishing an orientation relative to a defined orientation is explored.

3.2.9.1 Cue Displacement

The cue displacement will be determined in the shot selection algorithm. The shot selection algorithm already must determine the location of the end of the pool cue in order to verify a shot is reachable. With this information we are able to determine the point in space that the user must be located at. However we want to move the user along the edge of the table in order to simplify the guidance system. Therefore we will find the intersection of the table with the angle from which the pool cue must be shot. Our goal will be to then navigate the user until their pool stick reaches the desired location.

3.2.10 Test Cases

3.2.10.1 Game Modes

Billiards are a collection of games that are played with a billiards table, billiards ball, and cue stick. There are many different games played on billiards tables which include 8-ball pool, 9-ball pool, snooker, four-ball, cushion caroms, and many other variations of similar games. The goal of this project is not to implement all of these different billiards games, but rather to implement a working framework that can be expanded to different applications. For this project, a slightly modified version of 8-ball pool is implemented.

8-Ball Pool: 8-ball pool is one of the more common billiards games played because it is relatively simple and has fewer rules than many other billiard games. 8-ball pool consists of sixteen billiard balls. There is one cue ball, one black (eight) ball, one set of seven solid-colored balls, and one set of seven striped balls. There are two players who each will be assigned either solid or striped balls to try and pocket. Each player must use their cue to strike the cue ball in an attempt to push either the striped or solid color balls into the pockets. If a player sinks one of their game balls, they get to go again. If a player does not sink one of their balls it is the other player's turn. If a player sinks the cue ball or one of the other person's game balls, it is the other person's turn. If a player sinks the black ball before sinking all of the game balls, that player loses immediately. If a player hits the cue ball and does not hit any of their game balls, the other player gets to move the cue ball within a specified region.

The overall concept of 8-ball pool will remain unchanged in this project, but some small modifications are used to help with the implementation of the project. There will be one cue ball, one black ball, a set of three green balls, and a set of three blue balls. Reducing the number of balls on the table allows for less computation and a faster result for the user. It is reasonable to believe that the project can support more billiard balls at the expense of computation time. Sets of red and blue balls are used rather than solid and striped balls to implement a simpler computer vision algorithm. If the project was to use the standard solid and striped billiard balls, a computer vision algorithm that supports custom object detection would likely be needed. Like regular 8-ball, the player must hit the cue ball to pocket other balls. All other rules above are implemented except when the player cannot hit any of their game balls with the cue ball. Although this implementation is not a true 8-ball game, it is more than sufficient for visually impaired players. Figure 3.24 summarizes the actions supported by VISION.

Figure 3.24: 8-Ball Features Supported By VISION

The figure above summarizes the features supported by the project. Five possible events are being monitored, each event corresponds with a particular output. If the player sinks one of their game balls, does not sink one of their game balls, or sinks an incorrect game ball, the player will be notified and allowed to shoot again. If the player prematurely sinks the eight ball, they will be notified of losing the game. If the player sinks the eight ball after sinking all of their game balls, they will be notified of their victory. The results of every shot will be presented to the player and spectators on the visual display and audibly through speakers.

3.2.10.2 Shots Supported by VISION

The game of pool offers many shot selections besides the conventional straight shot. These different shots exist for several reasons, putting spin on a shot can give you better cue ball placement for the next shot, or a worse position for your opponent. A jump shot, in which you skip the cue ball over one ball in order to hit another is an advanced technique to give you a shot at an angle which no normal pool shot could have achieved. These various shots will be covered in this section in order to determine which will be kept and which must be discarded due to complexity. In order to simplify the distinction of shots, some shot types will be combined which more advanced pool players would recognize as separate shot types. This is due to the complexity of distinguishing between various shot types programmatically.

Straight shot: This is the most common shot where the cue ball has struck in order to directly hit one other pool ball. This is the main shot type which will be calculated. For simplicity this shot will include more advanced shots where the aim is to hit multiple pool ball in order to pocket a ball. VISION will support straight shots.

Bank shot: This is a more difficult shot which involves hitting the cue ball off of one of the rails (The walls of the pool table), and then hitting a pool ball. This shot type fits in with what is achievable within our simulation and shot selection algorithms and will therefore be kept. This shot will also encompass more advanced shots as long as the cue ball is hit off the railing. VISION will support bank shots.

Break shot: This is the initial shot which is taken to start the game of pool. There is not much that can be done to optimize this due to the random nature of the break. When that many different pool balls are placed right next to each other, small differences dramatically change the angles and forces of each ball. Therefore this shot will not be calculated. However it will still be used at the start of the game. VISION will not support break shots.

Jump shot: This shot is created to skip the cue ball over another ball in order to achieve a shot. Our simulation and shot selection algorithms will focus on the top down 2D aspects as proof of concept. We will therefore not be able to calculate this shot. VISION will not support jump shots.

Spin: This class of shot encompasses many types of shots. Spin can be used to make the ball go almost any direction after a hit as depicted in figure 3.25. This spin is achieved by hitting the pool ball in different locations and with different forces. While we could calculate side spin with our current model, we have decided that calculating spin will be difficult on our simulation as well as on the team responsible for directing the user on which location to hit the cue ball. We have decided to cut the added complexity of spin and instead focus on the basic concepts first. In another version adding spin will be of great benefit. VISION will not support spin shots.

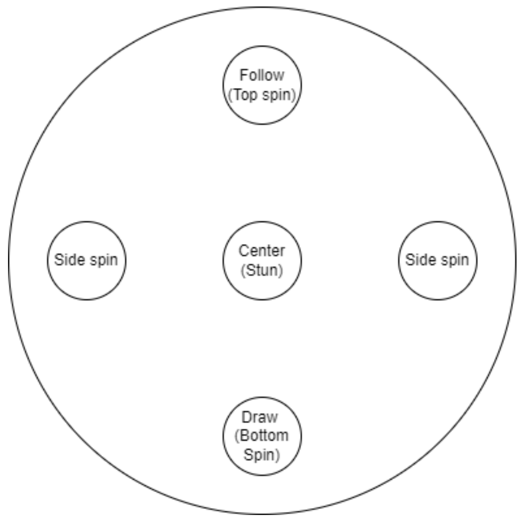


Figure 3.25: Cue Contact Point

3.2.10.3 Physical Limitations

These are constraints brought on by the physical limitations of the pool table, the pool cue, and the physical characteristics of the player. The simulations and shot selection algorithms are generally made for game type scenarios. This means that certain physical limitations are not taken into account. In this section we will discuss these obstacles and how we will attempt to overcome them.

Handedness of the user: This will factor in which hand a player uses to play pool. A shot which would be easy for a right handed player to shoot may be extremely awkward if not impossible for a left handed player. This difference is very large and could make a shot selection from the shot selection algorithm completely useless to the user. So even though this may be a difficult situation to account for, our solution will employ a system to insure that the shot given is a somewhat shootable shot for the player.

Length of the cue stick: This limitation ties in with the previous section on handedness. A shot in the middle of the table from the far end will be much too difficult to instruct a visually impaired person to hit. We therefore need a certain limitation on how far the to limit a shot's distance from the user to the cue ball. Giving a shot which the player cannot reach or that the user team cannot guide a player to will break the game and therefore must be accounted for in the shot selection algorithm.

Game balls in cue stick path: Shot selection algorithms for many pool games do not factor in the cue stick for a shot. In order to hit a straight shot there must be no pool balls in the path of the cue stick. We additionally need a small buffer for the players hand as scratching by accidentally moving a ball should be avoided where possible.

No available shot: If the shot selection algorithm is unable to find a safe shot to a pocket, there will be a few options:

- If the user has a ball which can be hit, the ball should be lightly tapped in order to avoid a scratch.
- If there are no good shots to hit one of the users game balls, the shot selection algorithm will respond with a shot that hits 3 railings of the pool table. This prevents a scratch.

White ball pocketed: When the white ball is pocketed, this is counted as a scratch. While we may be able to ignore other scratches, where the opposing player get an opportunity to move the ball, we cannot ignore this one as the ball must have a new placement. In this scenario, the user would have the option to place the ball down onto a certain section of the table, the user must then shoot in the direction of the far wall. This rule would require a completely new shot algorithm that specifically tends to this use case. Not only would the algorithm have to decide the best placement of the ball, but also must find the best shot in a certain direction. Adding this feature would create a lot of work for an occurrence which is not very frequent or important for VISION. Instead a simplification will be enforced. The ball will be placed at the same location that the break will occur and the player will also be allowed to shoot in any direction. Adding functionality for selecting placement and following the rules for a scratch will be very beneficial if not necessary for a competitive game. However, for this proof of concept we will instead use the simplified model put forward above.

Other scratches: In situations where the user scratches in ways such as, accidently moving a ball by means of something other than a shot, missing all game balls, or hitting a ball which is not yours first, the shot would normally be turned over to the opponent. In the case of our demo, we will instead be allowing the table to remain at its altered state. A new snapshot of the table state must be taken and a new shot selection must be made, the exact way in which the user will signify a scratch to the system will be taken care of by the SCRATCH team, but after that our system will treat the occurrence as any other shot.

3.2.11 Processing Unit

The computational needs of this project are intensive and require a powerful processor. The processor must be capable of performing artificial intelligence algorithms, computer vision algorithms, image processing algorithms, and various other types of general-purpose computing. For this reason, typical microcontrollers like an Arduino, ESP, or similar device will not suffice. The development boards that best suit the project needs are the Coral Dev Board, the Jetson Nano, and the Raspberry Pi 4 Model B. Although there are many other board offerings, the boards discussed in this section are some of the most highly recommended in the embedded computing community. Table 3.3 summarizes the technical specifications of the three major development boards under consideration.

Coral Dev Board: The Coral Dev Board is a small computer-like board designed specifically for machine learning tasks developed by Google. The board natively supports 2.4GHz and 5GHz wireless connectivity and Bluetooth 4.2. The board uses Mendel, a custom version of Debian Linux, so nearly all common Linux functionalities are available. Most importantly, the board has a built-in Google Edge TPU accelerator capable of 4 trillion operations per second. The board was specifically designed to run Google's proprietary embedded machine learning framework TensorFlow Lite. While the board has excellent performance for TensorFlow Lite programs, the board does not perform as well when trying to implement other types of machine learning frameworks.

Jetson Nano Developer Kit: The Jetson Nano is another powerful computer-like board designed for embedded machine learning applications developed by Nvidia. The board boasts its ability to run multiple neural networks at once to maximize all of its GPU cores. The Nano does not come standard with wireless connectivity or Bluetooth, so additional modules need to be added for wireless and Bluetooth connections. Nvidia utilizes a custom operating system, Linux4Tegra, on the Jetson Nano. Linux4Tegra is based on Ubuntu 18.04 so nearly all of the native Linux commands and utilities will be available on the Nano. Unlike the Coral Dev Board, the Nano is a more general-purpose computing device and can run Tensorflow, Caffe, PyTorch, Keras, MXNet, and many other machine learning software packages. Although the Jetson does not come with a machine learning Accelerator, the board is compatible with the standalone Google Edge TPU and can easily be integrated if desired.

Raspberry Pi 4 Model B: The Raspberry Pi line of microcontrollers is one of the most well-known in the embedded community and has a great reputation for being small, yet powerful devices. Unlike the other boards, the Pi was not developed specifically for machine learning tasks but rather as a small general-purpose computer. Despite not being designed for machine learning, the Pi is certainly capable of implementing smaller computer vision and artificial intelligence applications. The board comes standard with 2.4GHz and 5GHz wireless connectivity and supports Bluetooth 5.0. The Pi implements a custom operating system called the Raspberry Pi OS that is based on Debian Linux so it supports a majority of the common Linux features.

Processor	**Coral Dev Board**	**Jetson Nano Developer Kit**	**Raspberry Pi 4 Model B**
CPU	NXP i.MX 8M SoC (ARM Quad-Core)	Cortex-A57 (ARM Quad-Core)	Cortex-A72 (ARM Quad-Core)
GPU	GC700 Graphics Card (Vivante 16-Core)	NVIDIA Maxwell (NVIDIA CUDA 128-Core)	Broadcom VideoCore VI (Broadcom 4-Core)
RAM	1GB or 4GB	2GB or 4GB	1GB, 2GB, 4GB, or 8GB
OS	Mendel (Debian-Linux)	Linux4Tegra (Ubuntu-Linux)	Raspberry Pi OS (DebiDan-Linux)
Wi-Fi	2.4GHz and 5GHz	No	2.4GHz and 5GHz
Bluetooth	Yes (4.2)	No	Yes (5.0)
Ethernet	1GB Ethernet	1GB Ethernet	1GB Ethernet
HDMI	1- HDMI	1 - HDMI	2 - Micro HDMI
USB	1 - Type-A 3.0 1 - Micro-B 2 - Type-C	4 - Type-A 3.0 1 - Micro-B	2 - Type-A 2.0 2 - Type-A 3.0 1 - Type-C
Power	5V DC (USB Type-C)	5V DC (Micro USB or Barrel Jack)	5V DC (USB Type-C or GPIO)
Price	$129.99 - $169.99	$59.99 - $99.99	$34.99 - $174.99

Table 3.3: Summary of Processor Offerings

The table above summarizes the key aspects of the three boards. The most notable differences are in the GPU, Wi-Fi connectivity, Bluetooth connectivity, and price. The Jetson Nano has the most powerful GPU with 128-cores, significantly more than the other boards. The Jetson Nano is also the only board that does not come standard with Wi-Fi or Bluetooth connectivity. For a high-end development board, it is quite shocking that the board does not have any standard wireless communication features. There is a separate module for the Jetson Nano that includes Wi-Fi and Bluetooth 4.2 available for approximately $20 (Kangalow). The last major difference between the boards is their price. The price ranges of all the boards directly correlate to the amount of RAM chosen for the board. The price for each 4 GB board variation (assuming the Wi-Fi and Bluetooth adaptor is purchased for the Jetson Nano) is $169.99 for the Coral Dev Board, $119.99 for the Jetson Nano, and $99.95 for the Raspberry Pi 4 Model B. Despite having to purchase an additional module to have wireless access, the Nano appears to provide the most value among the devices. Table 3.4 (Franklin) summarizes the performance of the various development boards on common machine learning frameworks.

Model	Application	Framework	Jetson Nano	Raspberry Pi 3	Coral Dev
ResNet-50 (224×224)	Classification	TensorFlow	36 FPS	1.4 FPS	DNR
MobileNet-v2 (300×300)	Classification	TensorFlow	64 FPS	2.5 FPS	130 FPS
SSD ResNet-18 (960×544)	Object Detection	TensorFlow	5 FPS	DNR	DNR
SSD ResNet-18 (480×272)	Object Detection	TensorFlow	16 FPS	DNR	DNR
SSD ResNet-18 (300×300)	Object Detection	TensorFlow	18 FPS	DNR	DNR
SSD Mobilenet-V2 (960×544)	Object Detection	TensorFlow	8 FPS	DNR	DNR
SSD Mobilenet-V2 (480×272)	Object Detection	TensorFlow	27 FPS	DNR	DNR
SSD Mobilenet-V2 (300×300)	Object Detection	TensorFlow	39 FPS	1 FPS	48 FPS
Inception V4 (299×299)	Classification	PyTorch	11 FPS	DNR	48 FPS
Tiny YOLO V3 (416×416)	Object Detection	Darknet	25 FPS	.5 FPS	DNR
OpenPose (256×256)	Pose Elimination	Caffe	14 FPS	DNR	DNR
VGG-19 (224×224)	Classification	MXNet	10 FPS	.5 FPS	DNR
Super Resolution (481×321)	Image Processing	PyTorch	15 FPS	DNR	DNR
Unet (1×512×512)	Segmentation	Caffe	18 FPS	DNR	DNR

Table 3.4 Performance Results of Benchmark Testing

The table above shows the results of benchmark testing on common machine learning frameworks. Although the testing is done using a Raspberry Pi 3 rather than a Raspberry Pi 4, there is no evidence to show that the Pi 4 would have the massive upgrades necessary to outperform the Jetson Nano. The DNR (did not run) entries are indicative of the framework being too computationally complex, limitations in the hardware, or software that is not fully supported. The Coral Dev board performs really well when it supports the TensorFlow framework being used, but it does not support a wide range of frameworks. The Raspberry Pi and the Jetson Nano support a wide range of frameworks, but the Jetson Nano clearly outperforms the Pi across all of the benchmarks.

3.2.12 Communication Methods

Within the scope of VISION is the communication within VISION and the communication with the user side interface (the SCRATCH project team). The communication between these two will be minimalistic in nature to limit the effect of one project on the other. Key variables of interest would be transmitted via either wired or wireless forms of communication. Wired forms of communication are typically more reliable but will require the Jetson Nano (VISION team) and Raspberry Pi (SCRATCH team) to be located in close proximity to each other. Wireless communication is more advanced but is more common in practice. Wireless connectivity may be difficult due to the constraints of device communication on the UCF wireless network (UCF_WPA2).

Ethernet: Ethernet can be used to communicate between the Jetson Nano and Raspberry Pi. Each device can have a statically configured IP address and communicate over the ethernet connection. Both of the devices will be networked together but not be able to connect to any other networks. This approach is simple and reliable but limits the teams by not allowing either device to connect to the internet.

Serial Peripheral Interface (SPI): SPI is a very popular form of serial communication that can be used to interface microcontrollers with each other. SPI would primarily be used to establish a connection from the Jetson Nano to the peripheral ESP microcontrollers. SPI is not likely to be used to communicate with the SCRATCH team because this would require the teams main processors to be physically located together.

USB, USB-C, HDMI, and Other Common Connections: The Jetson Nano has a large port selection that can allow for may standard connections to be established. The VISION team intends to use these connections when possible to simplify the overall system. For example, the computer vision camera will be connected to the Jetson Nano with a USB connection.

Bluetooth: Bluetooth is discussed as a method for sensing user location, however, bluetooth is also a valuable option for data transmission of variables in the case VISION is looking to suit. Both teams will be using bluetooth for other transmissions and will have to ensure that the processors can support the number of bluetooth connections needed. There are many publicly available bluetooth libraries for Python that can be used.

Bluetooth can also be used to connect the Jetson Nano to the peripheral ESP microcontrollers.

Wi-Fi (TCP Connection): The Jetson Nano and Raspberry Pi can also communicate by establishing a TCP connection to each other and having a reliable communication stream. TCP is the ideal wireless communication protocol for this project because it is supported natively in Python, guarantees delivery of messages, and does not have a large latency. As mentioned previously, the viability of the TCP connection depends upon what the UCF network will allow. Preliminary testing shows that the UCF wireless network UCF_WPA2 does not allow for TCP connections to be established directly by devices on the network.

4. RELATED STANDARDS & DESIGN CONSTRAINTS

4.1 Related Standards

VISION needs to implement many technologies that have accompanying IEEE standards. Some of the most prominent technologies that will be used are Wi-Fi, Bluetooth, USB, micro USB, HDMI, UART, I2C, SPI, computer vision, machine learning, power supplies, Python, C, and cameras. These technologies have accompanying IEEE standards that will have to be further researched and documented in the official design document. This is not a fixed list and will likely change as further research into the project is completed. The main processor for VISION is a Jetson Nano, so many of the design decisions are based around compatibility and support on the Nano.

4.1.1 Wired Communication Standards

Universal Asynchronous Receiver-Transmitter (UART): UART is a serial data communication circuit that allows for variable data formatting and supports different transmission speeds. Most modern microcontrollers have a UART interface included standard in the serial communication integrated circuit. UART was invented by Gordon Bell of Digital Equipment Corporation in the 1960s (Digilent Corporation). Motorola, IBM, NXP, and other large corporations make a variation of a UART circuit that can be found in various processors and microcontrollers today. There is not a specific standard for UART but rather an agreed-upon format by chip manufacturers to ensure that the basic functionality of UART circuits is the same. The core functionality of different UART circuits will be the same across manufacturers, but additional features and implementation details may vary between manufacturers.

Impact of UART on Design: UART is a powerful communication method that is commonly used with microcontrollers to view output produced by the microcontroller. This can be helpful in debugging because many microcontrollers do not have a screen on them to view output. For this reason, the microcontrollers that are going to be connected to the Jetson Nano should support UART to allow for easier development.

Inter-Integrated Circuit (I^2C) Bus: I^2C is a synchronous, packet-switched, serial communication protocol that was invented in 1982 by Philips Semiconductors (known today as NXP Semiconductors). The most recent I^2C standard is UM10204. I^2C is free for programmers to use but does require device manufacturers to pay a fee to include the necessary I^2C pins on their devices. I^2C is primarily used for communication between chips on a single device. The widespread adoption of I^2C has led to nearly all modern microcontrollers coming standard with the necessary pins. The protocol itself is rather simple and only requires a serial data line (SDA), serial clock line (SCL), and ground (List). The protocol supports multiple controllers and targets (referred to as masters and slaves in some documentation).

Impact of I^2C on Design: Unlike UART which is used to communicate between individual devices, I^2C is used to communicate between chips on the same device. The main use of I^2C will be when configuring the microcontrollers. It is likely that I^2C will be used for reading from and writing to sensors. For example, the array of speakers will likely be written to using I^2C. The main design concern is ensuring that the microcontrollers chosen support I^2C.

Serial Peripheral Interface (SPI): SPI is a synchronous serial communication protocol that was developed by Motorola in the 1980s. Similar to I^2C, SPI is designed for short-range communication between chips on a single device. SPI is primarily used for communication in embedded systems and is supported by nearly all major microcontrollers. Like UART, SPI does not have a defined standard but its popularity has made it commonplace in the industry. SPI is so widely adopted than is it harder to find a device that does not support SPI that it is to find a device that does support SPI. SPI also makes use of the controller and target architecture but most SPI implementations only support a single controller. The SPI protocol requires four pins (a serial clock, MOSI (master out slave in), MISO (master in slave out), and chip select).

Impact of SPI on Design: SPI will likely be needed for communication on one of the microcontrollers needed for VISION. This will likely not impact the choice of microcontroller much because essentially every modern microcontroller comes standard with SPI support. The pins necessary for SPI communication are on just about every development board that can be purchased. The one problem that may arise is the different naming conventions some microcontrollers may use for describing SPI. The VISION team should be aware that some microcontrollers may implement SPI slightly differently and give the protocol a different name.

4.1.2 Wireless Communication Standards

Wi-Fi Standards: Wi-Fi has many standards associated with the technology but all stem from the IEEE 802.11 standard. The IEE 802.11 standard governs how nearly all wirelessly connected devices are supposed to function and must be strictly adhered to. The 802.11 standards were released in 1997 and continue to be amended as new advances in wireless technology are created. Although the standard has support for a variety of frequency bands, VISION intends to only use the 2.4GHz band. The 802.11 standards are specific to wireless communication while the 802 parent standard is more generic and involves ethernet connections as well. Both ethernet and wireless protocols will be needed for VISION.

Impact of Wi-Fi on Design: Although VISION will not be using Wi-Fi extensively as much of the project's functionality will be local, it will still be necessary for VISION to have wireless internet access. VISION will have a web page component that may be hosted over the internet so it is crucial the project can support such a connection. Additionally, the Jetson Nano will need to be configured over LAN before being connected to a Wi-Fi network, so the more general 802 ethernet standards will be used as well. VISION will follow all necessary 802.11ba (2.5GHz and 5 GHz) wireless communication standards.

Bluetooth Standards: The IEEE 802 class of standards also includes 802.15.1 which was the initial standard for Bluetooth communication between devices. IEEE no longer manages the Bluetooth standards and the Bluetooth Special Interest Group now manages the Bluetooth standard. The current Bluetooth standards require that a manufacturer's device meet specific requirements to market the product as Bluetooth. The widespread adoption and popularity of Bluetooth have led most devices capable of wireless communication to implement some form of Bluetooth. There are several companies that made Bluetooth modules specifically to allow devices to gain Bluetooth connectivity.

Impact of Bluetooth on Design: Bluetooth has emerged as the leading standard for short-range wireless communication between devices. It is assumed that if a device supports wireless communication, it will support Bluetooth (and Wi-Fi) at a minimum. The Jetson Nano does not come standard with wireless communication of any sort. However, the Nano does support a Wi-Fi and Bluetooth module in the form of a network interface card (NIC) that can be connected directly to the motherboard or inserted into a USB slot. VISION will need to obtain and implement either the NIC or USB solution to getWi-Fi and Bluetooth on the Jetson Nano.

4.1.3 Connection Standards

Connection Standards: There are many types of connections that can be established between devices such as GPIOs, USB, micro-USB, USB-C, HDMI, micro-HDMI, 3.5mm jacks, ethernet, DisplayPort, common wall outlets, and various other connection types. All of these different connection types have their own accompanying standards which must be adhered to. From a user perspective, many devices naturally support these connection standards. The VISION team will follow all standards and recommendations for connections based on the industry standards and manufacturer recommendations.

Impact of Connection Standards on Design: The main design consideration for common connections is ensuring that the hardware has enough ports available for all of the necessary components. Primarily, the VISION team needs to ensure that the central processing unit can support all of the needed peripherals. The Jetson Nano has a USB-C 3.0 port , a USB-C 2.0 port , two USB 2.0 ports , a USB 3.0 port , HDMI port, ethernet port, and 40 GPIO pins. Although there appears to be a large selection of ports available on the Jetson Nano, the VISION team needs to ensure that the port selection can accommodate all of the peripherals.

4.1.4 Programming Standards

Python Standards: Python's standard library is very extensive, offering a varied range of facilities such as built-in modules (written in C, others are written in Python and imported in source form) that provide access to different functions depending on the need of the user included but not limited to system operations working on both Unix, Windows based systems, or more specific programming functions used to solve everyday issues. Python for Windows includes the entire library as well as some additional components. On the other hand, for Unix like systems, Python comes in as a collection of packages, and additional packages or basic packages may need to be installed with the operating system to cover additional functions. The library also contains built-in functions and exceptions.

The latest release of Python is Python 3.10.7 released on September 05, 2022. Every release differs from the other by changing any of different syntax features, features in standard libraries or other customer libraries, typing and implementer features, or removing features, deprecating features, and restricting or removing restrictions.

Impact of Python Standards on Design: The Python standards are quite common and well documented. The VISION team will need to follow all suggested Python standards to ensure that their design functions properly. Deviating from the Python standards may cause undefined behavior in the program.

C Standards: The latest C standard is ISO/IEC 9899:2018, also known as C17 as the final draft was published in 2018. The biggest issue with using different standards is when a code returns a different output depending on the standard used by the code's compiler. The international standard which defines the C programming language is ISO/IEC 9899, a joint effort of ISO and IEC and the participating countries. The standard is then available for easy purchasing online. Each participating country adopts the standard into their own standards system while keeping the technical content the same.

Impact of C Standards on Design: The C standards have been around for a long time and are commonplace with the VISION team. The team will ensure to follow all C programming standards so that their programs function as expected. Similarly to the Python standards, if the team deviates from C standards, their programs may not function properly.

4.2 Design Constraints

4.2.1 Economic Constraints

The goal of VISION is to make a system that can detect billiard balls, plan strategic shots, determine the best position for a player, and localize and guide a user to the necessary shot position. The purpose of developing VISION is to broaden the inclusivity of societal pastimes to visually impaired individuals. With this in mind, the end user of this project is likely a visually impaired individual trying to play billiards rather than a company trying to make money off the product. The end user will likely have to fund the implementation of VISION themselves, so the project must remain as inexpensive as possible. After the project's completion, the hardware and software designs will be made available to the public, but users will still have to assemble some of the parts themselves.

For these reasons, the design must remain cost-efficient and relatively simple so that individuals of all backgrounds can implement VISION.

The components for the project were specifically chosen to meet requirements set forth by the Senior Design guidelines. For example, the Jetson Nano and accompanying Wi-FI and Bluetooth adaptor are needed as a central processing unit because the project must utilize an embedded processor. The software being developed for the project can be executed on any modern computer. An actual user can forgo the Jetson Nano and wireless adaptor for a laptop. This will allow a user to save hundreds of dollars, assuming the user owns or has access to a laptop. Similarly, a user that is interested in playing billiards likely has or has access to a billiards table. Not having to purchase a billiards table takes hundreds of more dollars off of the total cost to implement the project. By excluding two of the most expensive portions of the project that a user likely has already, the project is now able to be implemented for under $200.

The scope of the project is relatively large given the time constraints of the project and the team has not yet secured funding. To meet the goals of the project, artificial intelligence, computer vision, machine learning, location tracking, Bluetooth wireless communication, and many other complex technologies are needed. These domains each require some type of specific technology ranging from a few dollars to a few thousand dollars. VISION uses the least expensive technology that can still meet the needs of the project. Due to the project using cheaper technology, the accuracy, speed, and performance of the parts are somewhat limited. Careful consideration is used to ensure that the parts selected for this project will meet the requirements, while not being too expensive for a user to buy themselves.

4.2.2 Environmental Constraints

The VISION project is primarily going to be used indoors either in pool halls or different venues with billiards tables for visitors or in private residences for people who own their own pool table. Regardless of the location, one of the environmental constraints is to be weary of is the sound factor. Many systems in VISION rely on audio feedback to move the user around the pool table or to provide feedback via audio. Proper caution will be taken to make sure that the sound level is not overbearing for any user or those near the pool table. It is important that the sound provided stays audible and clear with minimal noise, and does not overlap when different systems need to provide audio feedback or instructions. One way we consider limiting these audio outputs is using only the speaker system as audio source so the user knows where to expect the sound from. This reduces distraction and focus from the central Jetson Nano controller that will be used to coordinate outputs. Our visually impaired user would only have to focus on sound coming from the speakers at set locations and the overall noise would only come from those areas.

Another minor environmental constraint would be aesthetics. Light or glare coming from the central monitor being used for our display or other lights coming off from the Jetson or other electronics. The monitor can always be set brighter or dimmer depending on the environment where the system is being used. For example, a dimmer pool hall using this system might want to keep the overall aesthetics dark and put off from using an overwhelmingly lit screen. The latter point itself is also be even more limited by encasing different electronics for their protection more importantly but also to reduce outgoing light.

A lot of the system's components can also be used and repurposed for other needs depending on the user. Our camera, localization aid, Jetson Nano and others can all be used modularly for other purposes offering the user additional options for reusing components if needed.

4.2.3 Social and Political Constraints

Billiards and social culture are inseparable in the societal domain. Constraints from this point of view should be examined as to allow for VISION to properly approach the social and political sphere. In terms of a physical social environment, an audio guidance oriented system may have limitations in its ability to be deployed. The proximity of audio output to the human ear can limit the efficacy of a guidance system significantly, and should be considered in both this prototype and in future design considerations thereafter.

The view of an assistive technology to the cultural and political masses primarily garners a positive view. Some cultural groups may look more highly on this system if they have a higher tendency or desire to play pool, and communities with impaired individuals will certainly find it a beneficial technological advancement. However, if the guidance mechanism is skewed to benefit one group over the other by means of a selected language being prioritized, this can lead to an inability for said group to be able to reek the benefits of the design.

4.2.4 Ethical Constraints

The main ethical constraint would be ensuring that the user's privacy is respected especially if the VISION systems are being used in pool halls where any number of people would end up using the product. The camera system should not be used to record any user, player, or individual in the vicinity of the table. The camera system will be pointed above the table at all times and will be primarily used to detect the balls still in game as needed for computer vision purposes.

Any other recording apparatus would only be used in the closed system that is our product and will not be relayed through any other means. Communication between the VISION team and the SCRATCH team for our dual project will be done through a secure Bluetooth socket, limiting interference and increasing privacy if either were to be a concern for the user.

4.2.5 Health and Safety Constraints

When new technologies seek to assist visually impaired individuals, the safety of the user is priority one. Creating a device that harms rather than helps a user is the worst case scenario, and must be considered to make sure a design is an additive to the lives seeking assistance. Constraints of VISION in this regard stem primarily from the navigational system in place. Navigating a table with limited awareness of surroundings can easily lead to a user tripping over scattered or loose items. In the case of VISION, the apparatus being used to hold up the camera is a constant obstacle that must be considered and shifts in design made with this obstacle in mind. Additionally, prioritizing a lack of exterior obstacles as well as carelessly placed design components will lead to a safer navigational path.

In any project including electrical components, proper insulation and safety measures for all components must be considered to prevent the user from any chance of electrical shock. Additional electrical signals in audio that are used for output guidance should be in a form that is also safe for the user in both electrical contacts and auditory capacity. For instance, proper frequency and signal shapes as well as volume can prevent damage to hearing for users that rely on this ability. Any localization method should be utilized in a manner that also does not put the user at risk of harm.

4.2.6 Manufacturability Constraints

One of the biggest manufacturability constraints is the availability of the parts that we will need, especially the Jetson Nano. We will make sure to get one ahead of time, or keep in mind the possibility of market shortage and take into account other options that are suitable. Beyond this, our choice for other systems will also have alternatives either from the same company or from different companies in case our expectations do not match the product acquired or in case of any unexpected supply-chain shortage.

We may also be constrained by resources or skills for how we would want to encase, wire, or prop up different components. For instance, we would have to consider different options for lifting the camera above the table in a way that is feasible within our means and easily transportable when the whole system would have to be moved. Ideally, the table, speakers, localization aids or beacons, and the camera system would all be moveable as a single system. Wiring from these systems to the Jetson Nano or main controller is also another concern, as it would have to be flexible enough to not be an issue for anyone moving around the table. For the encasing scenario, we will have to envision different options such as buying readily made cases to protect the Jetson Nano or microcontroller, doing some woodwork, or 3D printing.

4.2.7 Sustainability Constraints

The system should be designed for long-term use. The VISION system has a good mix of battery-powered devices and wired devices that both incorporate additional constraints in our system. The battery-powered devices such as the beacons should either have enough usage to last for a year while being constantly turned, or have an option for the user to easily and safely turn them on and off as needed without affecting the software portion of the project. Other battery-powered components should follow a similar or better lifetime cycle.

The central computer (Jetson Nano) might also be susceptible to different issues as any computer would be. Careful consideration will be made to ensure that all the computationally intensive portions of the system running on the Jetson in parallel do not exceed the processing power of the Jetson. Also, different files will be stored on the Jetson and used for the display introducing the need to make sure that file storage is taken into account and properly monitored for the whole system.

Other systems that are powered via wiring from outlets would also introduce constraints on power consumption for the user, as well as issues with heating where applicable. The total system should not introduce any power surges or heating surges to the user.

5. SYSTEM HARDWARE DESIGN

This section goes into the details on the hardware design of the entire integrated system. As the research section dove into the various components of the system and how they plan to facilitate the goals of the design, this section discusses the specific components to realize those goals and the manner in which they will interact with one another and be connected.

5.1 Billiard Table

From pool halls to at home setups, billiards tables come in a range of shapes and sizes. Determining a table that best meets the desired needs of the project is crucial to the mapping of the remainder of the design. Considerations for this selection range from ease in mobility of the table, sturdiness, ability to facilitate all subsystems and adaptations, robustness to case testing common occurrences, and ease of display for showcasing purposes.

The standard for billiards tables includes six pockets and is in a rectangular orientation with two pairs of matching sides at a 2:1 length ratio (Roeder). Tables come in four standard size orientations as followed (Vudrag):

- Standard - 8ft x 4 ft dimension. This size is commonly used by at home and beginner setups. It has enough space for complex shots, while not requiring too much power to practice basic shots.
- Large - 9 ft x 4.5 ft dimension. This size is the recommended professional orientation as it requires more physical skills to move balls to desired locations. Certain shots are more challenging with greater distances, such as when balls are in close proximity. Beginners have been shown to struggle on this type of table
- Bar Box - 7 ft x 3.5 ft dimensions. This orientation is preferred by some for its ease in ability to make shots, allowing it to be a popular orientation for social settings. Several common issues springing up from the use of this type of table include: tough to reach pockets, poorly matted felt, dead rails, and issues relating to cue ball size. Clustered groups become more common in this setting and create a more luck based game compared to skill focused playthrough.
- Miniature - This table orientation encompasses tables ranging in sizes of the longer length from 20 inches to six feet. These sizes are commonly used for tabletop billiards or by children. Rooms with limited space will possibly be a proper fit for an orientation such as this as well. These sizes are not expected for use in a serious game of pool.

In respect to VISION, the proof of concept aspect of our project and the augmented scale of the game that is planned to be deployed is best performed at smaller orientations of size. The scale of the table also positively correlates with price, so a smaller orientation table will best suit our endeavors. While the large orientation is quickly ruled out, bar box and standard orientations would be favored in the case of an at home asset for appearance. Miniature tables however are the preferred for the project's use case as it is a heavy favorite in terms of portability, lower price, and ease of creating augmentations to the minimalist approach many of the tables in this category have. An additional benefit of this orientation is the opportunity to develop the project on a folding billiards table. This type of table would be accompanied by the asset of mobility to easily transport it within a team member's car for presentations and development of the prototype project.

Several suppliers can facilitate a table as specified at a range of prices and specifications. Two tables of interest meet the criteria of lower size and foldability from the suppliers of Blue Wave and Rack as shown in Figure 5.1. These are comparable models, with the Blue Wave model being of higher quality, dexterity, and price to the half-priced Rack model. Additionally, both are miniature tables at sizes of 72 inches and 55 inches respectively.

The Fairmount model was chosen for the final design. Initially, the Rack model was going to be used, but upon realizing the smaller size constraints included smaller balls and a noticeably detrimental impact to game performance, the larger table was chosen for use in VISION.

Figure 5.1: Blue Wave's Fairmount Table (Left) & Rack's Crux 55 Table (Right)
(Awaiting Permission from Blue Wave and Rack)

5.2 Camera

5.2.1 Computer Vision Camera

The computer vision section of this project is responsible for obtaining an image of the current state of the billiard table, identifying the cue ball and its location, and determining the location of all of the other billiard balls in play. The computer vision algorithms rely on a high-quality image of the table state to be able to process the image and extract the necessary information. The camera should be able to be mounted above the table, take clear pictures of the table in a variety of lighting conditions, have a wide field of view, and be compatible with the Jetson Nano.

The camera will take pictures of the billiard table that will be processed by computer vision algorithms. Higher quality images will be able to provide better contrast between the background and the billiard balls of interest. To ensure the best results, a camera that provides a video resolution of at least 2 megapixels is desired. If a lower resolution is needed by the image processing software, it is possible to reduce the resolution to what is needed. However, it is not possible to exceed the maximum resolution of the camera. For this reason, the safest option is to get a high-resolution camera and scale down the resolution if needed.

The field of view of a camera describes how wide of an angle a camera can view. A field of view corresponding to 60° would only see a small portion of what is in front of the camera while a field of view of 180° would see everything that is in front of a camera. A larger field of view allows for the camera to be positioned closer to the billiards table. Most webcams have a field of view of 60° - 90°. The ideal field of view for this project is around 90°. A field of view of 90° will allow for the camera to be mounted about a meter above the billiard table and still be able to capture the entire table (Pinke).

The Jetson Nano supports a wide range of camera interfaces including MIPI CSI, Ethernet, FPD-Link III, GigE, GMSL, PoE GigE, USB, and V-by-One HS. Of these interfaces, Nvidia recommends using a MIPI CSI or USB interface because these options are supported natively (NVIDIA Corporation "Taking your first . . ."). Additionally, both of these camera types can provide high-resolution images at an affordable price.

Summary of Requirements:

- Camera can be mounted above the billiards table
- Have a minimum video resolution of 2 megapixels
- Provide a field of view of approximately 90°
- Utilize an interface supported by the Jetson Nano
- Does not exceed $100 in price

<u>*MIPI CSI Cameras:*</u> MIPI is an alliance of large technology companies that develop specifications for devices in the mobile-computing industries. One specification defined in the MIPI standards is the CSI-2 (Camera Serial Interface - 2) which has quickly become one of the most popular interfaces for implementing cameras in embedded designs. CSI-2 is a high-speed protocol for sending images and video from a camera to a computer via a proprietary MIPI CSI connector.

In recent years, CSI-2 cameras have become the clear choice for many embedded processing applications. With the creation and wide-scale adoption of the CSI-2 protocol, many large electronics manufacturers have started manufacturing CSI-2 cameras leading to a wide variety of options in the market. For this reason, these cameras are relatively affordable and there are many options available for $20-$30. Furthermore, CSI-2 cameras provide higher bandwidth for pictures and images at a price comparable to USB cameras of much lower quality.

One of the most commonly used CSI-2 cameras for embedded applications is the Raspberry Pi Camera Module V2 which offers an image resolution of 8 megapixels and full HD video at only $25 (Raspberry Pi). The high performance at low cost is what makes CSI-2 cameras so popular. The main concern with the Raspberry Pi camera, and other CSI-2 cameras, is the short cable length of the camera connector. CSI-2 cameras typically have a maximum cable length of 20-30 cm.

The short-range of CSI camera cables means that the Jetson Nano will have to be located next to the camera. Having the Jetson Nano next to the camera may not be possible based on the mounting location of the camera. The camera needs to be mounted above the billiards table facing downwards so that an image of the current state of the billiard balls can be captured. Having the Jetson Nano mounted above the billiards table would not be ideal because all of the other project components would have to have interface with the Nano in a hard-to-access location. Due to the limited length of connections for CSI cameras, it is unlikely that one can be used for this project.

USB Cameras: The next best alternative is to use a USB camera. USB cameras are natively supported by Jetson Nanos and are one of the camera interfaces recommended by Nvidia. Although the performance of USB cameras is not as high as a CSI camera, most USB cameras are suitable for the project requirements. Using a USB webcam will not require the Nano to be mounted directly next to the camera, allowing for the processor to be located in a more centralized location.

Many USB cameras will meet the requirements. It was determined that a moderately priced webcam would meet all of the requirements and nearly all webcams are USB devices. Many different webcams from reputable suppliers were considered. Four selected webcams that best meet the required specifications are summarized below. Any webcams that are not readily available for purchase or greatly exceed the budget requirements were not considered. Table 5.1 summarizes the specifications of the highest recommended web cameras within VISION's budget.

Camera	Manufacturer	Price	Resolution	Field of View
PowerConf C200	Anker	$69.99	2K	68° - 95°
PowerConf C300	Anker	$129.99	1080p HD	78° - 115°
C920s Pro Full HD Webcam	Logitech	$69.99	1080p HD	78°
C930s Pro HD Webcam	Logitech	$129.99	1080p HD	90°

Table 5.1 Summary of Camera Options

From table 5.1, the Anker PowerConf C200 is the best choice for the computer vision camera. This webcam is one of the cheapest cameras that not only meets but exceeds the project requirements. The camera has a video resolution of 2K, which is better than the 1080p resolution that the other cameras have. The camera also has three field of view angles: 65°, 78°, and 95°. The ability to use different field of view angles will be helpful when testing the design to find a camera height and angle that allow for the clearest pictures to be taken. The PowerConf C200 also supports autofocus and low-light environments to capture the best possible image regardless of the conditions around the billiards table.

5.2.2 Computer Vision Camera Mounting

To capture an image of the billiard balls, a camera will be needed above the billiards table. The camera can either be fixed to the ceiling of the room where the billiards table is located or mounted to a structure that extends over the billiards table. Ease of access, portability, and reliability should all be considered when selecting how to mount the camera above the billiards table.

Ceiling Mounted: Having the camera mounted to the ceiling of the room is appealing because there would be no obstructions to the billiards table. This is ideal because players would not have to maneuver around a structure and possibly have to alter shots due to the camera stand being in the way. However, this implementation would not allow for the billiards table to be easily moved between locations and limit where the system can be implemented. Furthermore, if the camera is mounted at different distances above the table, the computer vision algorithms being used may need to be revised to account for the changes in distance.

Fixture Mounted: Another possible way to mount the camera is to create a semi-permanent fixture that extends above the billiards table. Such a fixture would allow for the camera to be mounted above the table regardless of the table's location and is shown in figure 5.2. This solution would also allow for the entire system to be transported between locations without having to mount a camera on a different ceiling. This approach will also make the computer vision algorithms more reliable because the distance from the camera to the billiards table will be fixed regardless of where the system is being used (Pinke).

Using a fixture to mount the camera above the billiards table seems like the better solution because the billiards table will need to be mobile to some extent. As of now, the billiards table does not have a permanent location. Being able to move the table without having to recalibrate the camera, modify the computer vision algorithms, and remount the camera to a ceiling are all important factors for developing the system. The structure will only need to support a small webcam and can be made small in comparison to the table. When the camera structure is made, priority will be given to minimizing the structure size to have as small of an impact on the billiards table as possible.

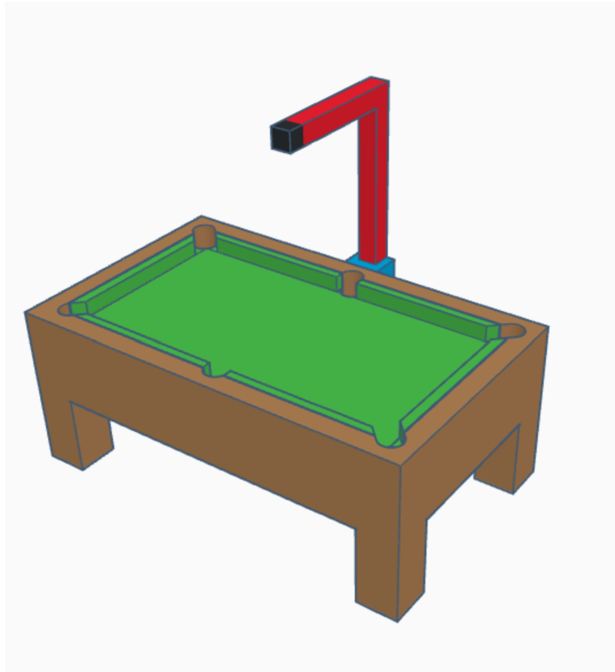


Figure 5.2: Example of Fixture Mounted Camera

5.3 Visual Display

The visual display system is responsible for showing the user dashboard. The dashboard will show the current shot selection as well as the various game stats collected by our system. A typical LCD monitor will suffice for VISION. The team has opted to use their own monitor that has an HDMI input to be compatible with the Jetson Nano.

5.4 Localization System

Based on the different options presented in the research section, VISION has decided to initially fully focus on Bluetooth Energy as the localization scheme and navigation scheme. The system will navigate the user around the pool table, from their initial position to the target position for optimal shot computed by the pool game algorithm along a path determined by our navigation algorithm. In essence, the plan is to compute the user's localization at every point using trilateration. When the system gets input from the user that they are ready to make their next shot, a series of actions begin to allow us VISION to determine where the user is around the table at the current time.

BlueCharm Beacons: Three beacons will be placed on the pool table at specifically chosen locations. The beacons will be sending out advertisement packets at the smallest possible interval in order to get the best accuracy. A possible option for beacon locations is shown in figure 5.3 for a regular pool table of length 2.54m horizontally and height 1.27.

Figure 5.3: Pool Table with BLE Beacons

Our choice of beacon will be between the BC-U1 or BC-08 from the BlueCharm. The main difference, besides their structural/appearance, is that BC-U1 is USB powered versus BC-08, which is battery powered. This matters in the sense that the USB powered beacon would allow us to broadcast at very low intervals of about 100ms minimum or about 10 transmissions per second. As described earlier, a better accuracy will be obtained by averaging the RSSI values obtained throughout the course of scanning. Being able to obtain multiple transmission values would then be extremely valuable for our application. BC-08 is advertised as being their product with the best battery life, which is a considerable advantage since broadcasting at these very small intervals drains the battery life faster. Picking BC-U1 over the BC-08 will remove the concern around battery life but will introduce an additional concern over how to power three different beacons around the pool table in an ergonomic manner.

ESP32: The scanner in this case will be an ESP32. The ESP32 is a low-cost, low-power system on a chip with two processors capable of both Wi-Fi and BLE capabilities. The overall picture is that the user will be holding onto a case with an extruding button that they can press to determine when they are ready to start their next shot. Once the button is pressed, a code will run on the esp32 which will read the advertisement packets from the three beacons and send out the results via Wi-Fi to the Jetson Nano. During this period of time, the user will have to stay in position to get accurate results. The ESP32 will thus be pre-programmed beforehand to contain the code capable of recognizing a button push, reading the RSSI values and sending that information to the Jetson Nano.

Due to the need to keep the ESP32 in a constant mobile state, it will need to be powered through a battery present in the case containing the ESP32 and the connected button. Table 5.2 outlines the power consumption in the ESP32 ((“Insight Into ESP32 Sleep Modes & Their Power Consumption”)):

Mode	Description	Current Consumption
Active mode	Keeps everything running including the real time clock, peripherals, Wi-Fi and bluetooth modules and the processing core and coprocessor	80-90µA for Wi-Fi or Bluetooth receiving and listening
		120mA for Wi-Fi or Bluetooth transmitting at 0dBm
		160-260mA for Wi-Fi or Bluetooth transmitting at 13 to 21dBm
Modem sleep mode	Processing core, real-time clock, coprocessor active. Wi-Fi module, Bluetooth module and peripherals inactive	3-20mA
Light sleep mode	Real-time clock, coprocessor active. Wi-Fi module, Bluetooth module and peripherals inactive	0.8 mA
Deep sleep mode	Real-time clock, coprocessor active. Processing core, Wi-Fi module, Bluetooth module and peripherals inactive	10µA
Hibernation	Real-time clock active. Coprocessor, Processing core, Wi-Fi module, Bluetooth module and peripherals inactive	2.5µA

Table 5.2: Summary of Power Consumption in ESP32

The software portion of the localization scheme will take into account these different options and allow power savings by having the device go to a specified sleep mode when the user is in between shots and wake up from said sleep mode to allow for Bluetooth and Wi-Fi capabilities to turn on and function as described above.

The ESP32 operates between 2.55V - 3.6V. This is the deciding factor to determine what batteries to use and how many batteries are needed. Our choice of battery must also be able to deliver the needed current consumption for Wi-Fi functionalities described above.

NiMh batteries delivering about 1.2V each would not be enough unless 3 of them are being used which seems harder to design for. Lithium polymer batteries providing about 3.7 to 4.2V would be too much as well since a lot of the energy would be used to bring down the voltage to usable levels. Lithium batteries providing about 1.5V each can be used perfectly. A constant 3V can be obtained from 2 of them. For example, Energizer AA Lithium Battery are advertised to provide a low discharge rate which would make sure that the voltage remains at a steady 3V for a longer period of time and have a longer energy density than NiMh or NiCd batteries and at 500mA of current, it has a capacity of 3000mAh which would correspond to powering our device for 6 hours straight as compared to about 3 hours for alkaline batteries for instance. The ESP32 will not be drawing that much current constantly either way, but this would ease or solve our concern for WiFi functionalities.

To summarize the ESP32 will be battery powered to two of the above batteries in a battery holding case and soldered to a button connected to GPIO pins on the ESP32 configured to detect any button presses. The total set up will be in a 3D printed case. There are different .STL files online that we will only modify slightly to adjust to our system requirements such as having an opening on top for the button, and possibly holes on the side if the case is built to be worn as a necklace, or a single hole with a buckle for other fitting options.

Jetson Nano: The Jetson Nano will receive the RSSI values from the three different beacons and differentiate them depending on the IDs. A Python code will then run on the Jetson Nano to compute through trilateration the x and y coordinates of the user with respect to the bottom right or left end of the table, which will be our (0,0) coordinates in this case. The (x,y) coordinates will then be sent off and used for our navigation or guidance algorithm.

This overall process will be repeated for each shot whenever the user presses the button and determines he’s ready to be guided for his next move. A few concerns about this approach, as outlined in the BLE section, are due the accuracy of the results. Due to this, we performed some testing on one of the Bluecharm beacons with the ESP32 to test variability in the RSSI values under different conditions, computing the distance between the two based on the formula mentioned earlier with different environmental n values.

5.5 User Guidance System

At the heart of the goal of VISION is the ability to guide an impaired user to a desired location on the table and allot them the opportunity to make desired shots. The method for achieving this guidance must have solid logistics, be reliable within worst case board states, and be safe for the user’s traversal of the table. The following outlines the methodology to accomplish this and the specifics of the design that minimize unwanted circumstances within gameplay.

5.5.1 Audio Array Design

The two primary methods discussed in the technology review conducted in section 3.2.5 on guidance relied on audio and haptic feedback. Haptic feedback is revealed to be a great technology in tandem with other devices to create a detailed picture for users in dynamically changing environments. However, for the static pacing of VISION that includes a necessity for directions around a stationary table and angular orientation relative to it, the limited information delivery that can be done by haptic feedback is a hindrance. Moreover, an apparatus on the user would be required for the navigation around the table, which would add more complexity to both the easy use of the system and the SCRATCH team's present user system. This system also would have flaws in communicating coherent instructional guidance and would require a feedback loop for validation of positioning of the user.

On the other hand, audio guidance can be deployed in a rather convenient manner that comes with several advantages. With the use of several small speakers around the table edges in an array fashion, guidance algorithms can pinpoint the desired path for the user to take around the table for a designated shot. This can be accomplished with an updating location of the user being referenced for the proper speakers to activate to give an accurate route for the user's destination. Once in position the array can then be turned into a angular guidance system to orient the user within a margin of error of the ball to then hand off to the user team for finer user mechanics.

To properly distribute the necessary signals to a single desired speaker at a time, the Jetson Nano will be the primary decision maker that will communicate signals via SPI to an ESP32. This ESP will interpret the data on speaker activation and then select the proper speakers to be activated by use of a demultiplexer that is able to select a singular output via digital selection pins. To access upper levels of volume, the output signal will be integrated with an audio amplifier from the ESP. A prototyped singular speaker design is shown in Figure 5.4, showing an example of how an ESP32 can communicate the described outputs. Navigation algorithms described in Section 6.3 explain how the Jetson will comprehend speaker choices. Once the ideal position and orientation are reached, signals sent to the ESP will stop until further navigation is desired. The output signal will consist of a fluctuating PWM square wave with a 50% duty cycle that jolts on and off every half second. This allows for an easier to locate and orient origin point given the differential signal.

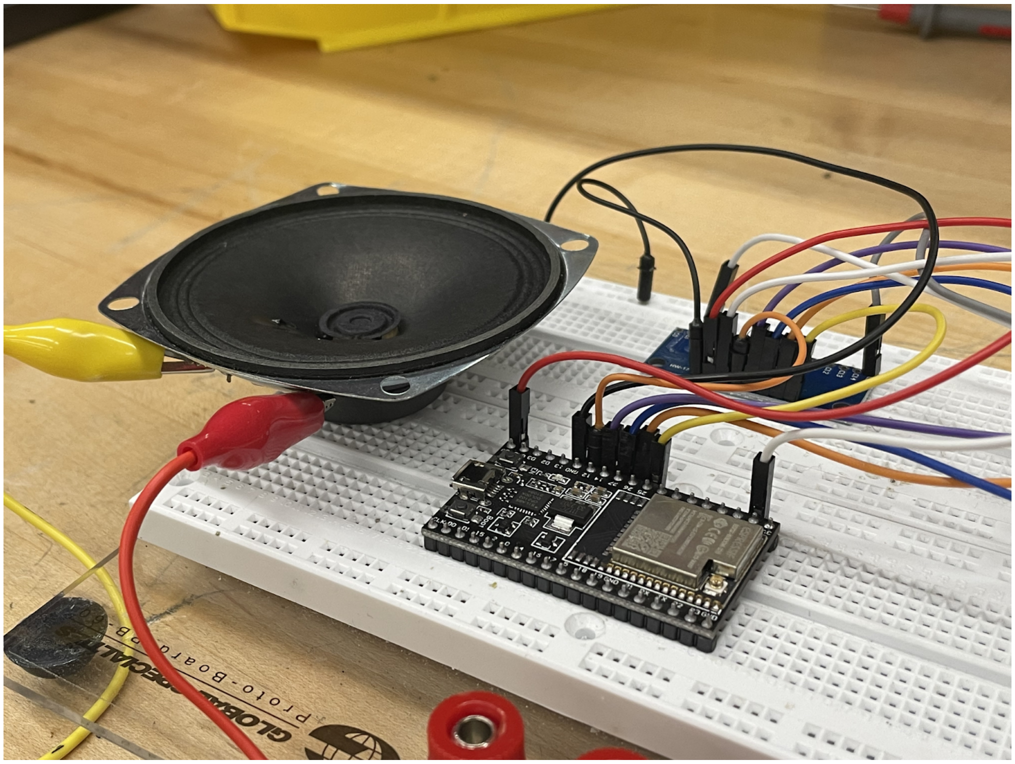


Figure 5.4: Prototype Speaker Activation Design

The specified positioning for the speaker array in VISION will include 12 speakers at the perimeter of the table as shown in Figure 5.5. This method allows for the positioning guidance goals of VISION to easily be attained, and gets the orientation parameters within an acceptable margin of error as described in Section 5.5.4. Each speaker is approximately 19 inches apart.

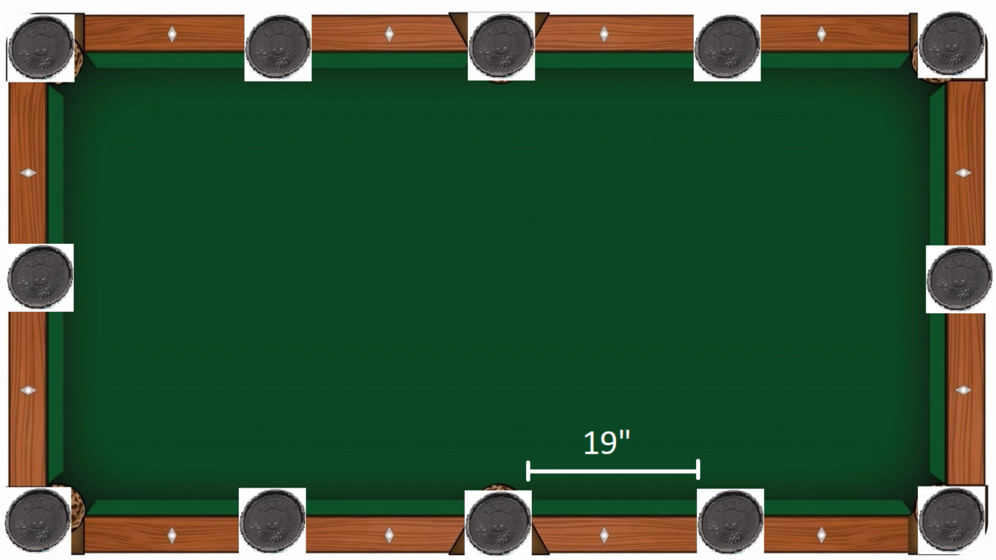


Figure 5.5: Designed Speaker Array

5.5.2 Positioning Method

Navigation of the impaired user will rely primarily on audio guidance from VISION's table speaker array. In the case of positioning, corner speakers will be activated to best guide the user along a 2D plane that consists of only two possible directions to the user. In any instance of user location, a speaker on the corner of the table will be activated with the user having knowledge to walk in the direction of the origin point of the sound. Upon reaching the desired location, the speaker will cease to output sound or will output from an alternative location if in an improper location.

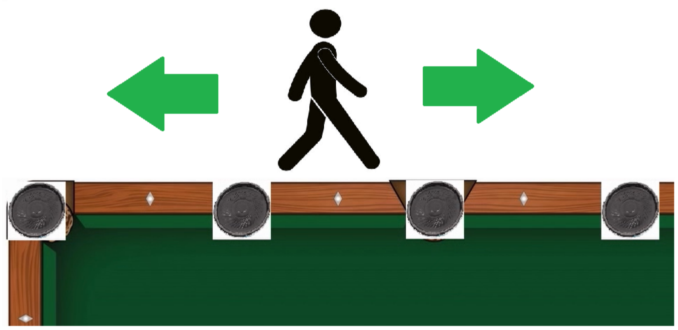


Figure 5.6: Bidirectional Guidance Possibilities

5.5.3 Orientation Method

Upon reaching the desired location around the table, the intermittent speakers around the table now are used to orient the user to an approximate location that places them in line with the cue ball and the direction in which to shoot. Since the orientation mechanism lacks an active feedback method, the orientation speaker will play for a 10 second period to give the user ample time to shift position.

This mechanism being the case does leave a possibility for a variable margin of error for the user. The calculated worst case angular margin lies at 7.1 degrees with a maximum possible arc difference of 8 inches. These values are within the 15 degree worst case scenario proposed in VISION's project requirements, and allows for a viable hand off to the SCRATCH project for fine tuned movements. Figure 5.7 further shows the worst case margin of error scenario. Additionally, locational accuracy may also introduce added margin of error that must be smoothed out for most cases and troubleshooted for higher accuracy to give a possible starting point to the SCRATCH design.

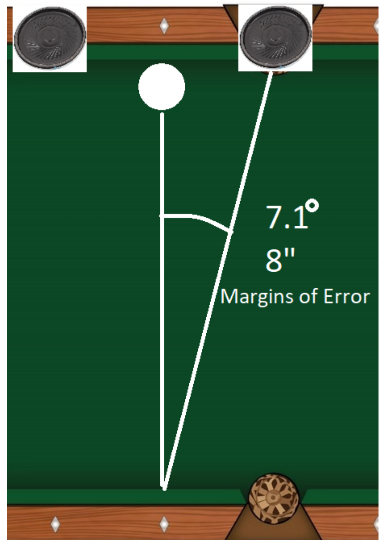


Figure 5.7: Worst Case Margin of Error Estimation

5.6 User Control Interface

To properly control the full array of controls VISION's system requires, custom user interfaces are designed to relay critical commands to the system. The section on user commands outlined three possible command interfaces for our design, including a remote control on the user, centralized control on the table for an assistant, and an audio command interface. As this project is a proof of concept, the simplest command interface will be integrated in a centralized command interface for an assistant to perform necessary commands. The interface will be minimally invasive to the action within gameplay, and will largely be for a short list of commands that are integral to procedural operations of VISION.

There will be four push buttons that will be integrated to the ESP and PCB of the project. These will include commands for starting, pausing, and stopping game play as well as relaying that a turn has finished. The first three commands are integral for the usability and ease there of for the player, and the latter is important for allowing the system to comprehend a finished turn and calculating the next move. The computer vision portion of software is not constantly updating, requiring a need in this model for some sort of action to change the code's operation mode.

5.7 Communication Network

The communication network for our hardware will allow the different computing systems to handoff information and control with the correct timing. Ensuring our purchased hardware is compatible with the protocol discussed below is another key factor in making sure we build a successful communication network.

5.7.1 Communicating Systems

The following subsystems must be connected for our system to work properly:

- Computer Vision
- Shot Selection
- Display Software
- Table Feedback
- User Localization
- User Guidance
- User Control Interface
- User Team System

Some of these systems will be present on the same hardware, while others will require some form of communication protocol to receive necessary information. We will now look how these systems are separated onto different hardware, as well as what hardware must communicate with each other. The computer vision, shot selection algorithm, and display will all be on the Jetson Nano. This will leave the communication between these systems as a software design specification. Table feedback and user guidance will share a microcontroller. The table feedback and user guidance systems both require communication from the Jetson Nano. The Jetson Nano will communicate with the microcontroller using SPI. The user localization system needs several pieces of hardware to function properly. It needs the beacons, the scanner (ESP32), and the Jetson Nano for calculation. The Jetson Nano will connect to the ESP32 via Bluetooth. The Jetson Nano will act as a server and the ESP32 will act as a client. The ESP32 will connect to the beacons with Bluetooth. The ESP32 will act as a server and the beacons will act as clients. The user control interface will require two pieces of hardware, the transmitter and the receiver. The control interface will connect with the Jetson Nano via Bluetooth. The Jetson Nano will act as a server and the control interface will act as a client. The user team will receive all needed information through one Bluetooth communication line connected to the Jetson Nano. This will reduce the coupling of the systems, which is generally best practice. The Jetson Nano will act as a server while the user team's processor will act as a client.

Figure 5.7 displays the hardware for each system as well as the needed lines of communication. While most algorithms are not shown on this diagram, the localization algorithm was kept to show the user localization system and why the system includes the Jetson Nano.

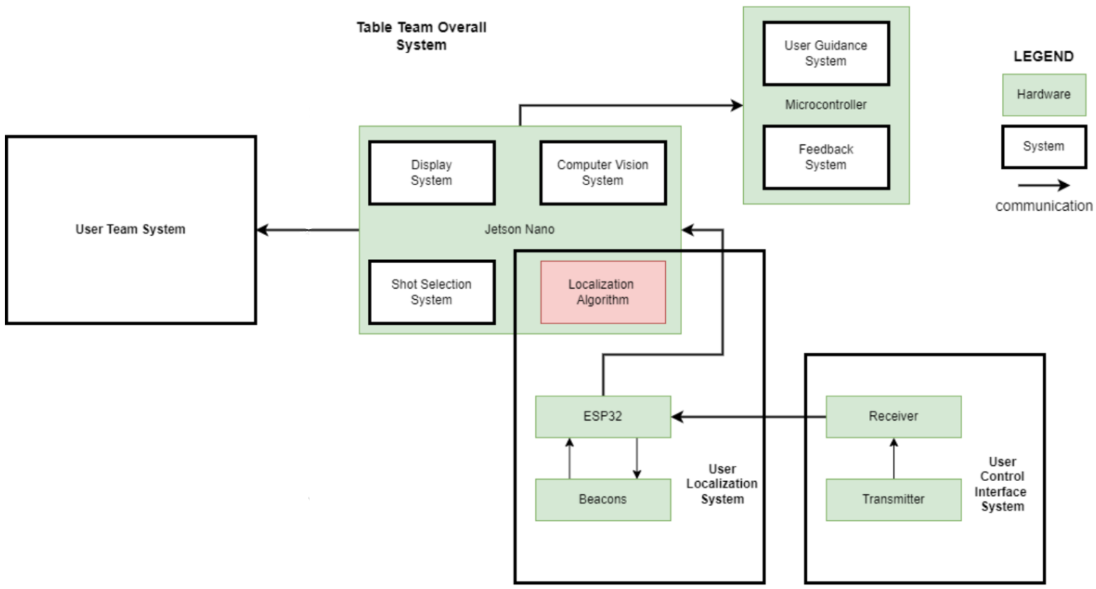


Figure 5.8: Tentative Communication Network

5.7.2 Communication Protocols

Event vs State Driven Communication: It can be hard to define the VISION network into event or state driven. As described in (Rollins). While there is an event driven process controlled by the User Control Interface, this is a one time action which places the system into a state, such as paused or in play. Looking over this the team has decided to treat the system as an event based system, this is because it will go dormant without user interaction. The user must alert the system that a shot has been taken to start the process of reading the table and generating the next move.

Processor Communication Capabilities:
Table 5.3 summarizes some of the relevant processors and what types of communication protocols they have access to.

Processor	I2C	UART	SPI	Bluetooth	Wi-Fi	Ethernet
Jetson Nano	4	3	2	No*	No*	Yes
MSP-EXP430FR6989	2	2	4	No	No	No
ESP32	2	3	3	Yes	Yes	Yes

Table 5.3: Comparison of Communication Interfaces

*Note: the Jetson Nano does not have Wi-Fi or Bluetooth connectivity by default, but can gain access to these forms of wireless connection with an adapter.

From the chart it can be seen that there are many available wired connections for communicating between the Jetson Nano and the MSP-EXP430FR6989. However, there is an issue with the Jetson Nano communicating with the ESP32 over a wireless connection. With the standard Jetson Nano there are a couple of options we could take for wireless communication.

- Connecting the Jetson Nano to ethernet and the ESP32 to WiFi. The two could then make API calls over the internet
- Setting a second proxy ESP32 in a wired configuration to the Jetson Nano, then communicating through bluetooth or WiFi with one ESP32 to the other.

These two options are possible but would be more complicated than getting a Wi-Fi or bluetooth adapter for the Jetson Nano that would allow for direct communication. An example would be the Intel Dual Band Wireless-Ac 8265 w/Bluetooth 8265.NGWMG along with an antenna that can support both 2.4 and 5Ghz. The suggested antenna from a tutorial suggests using a molex film antenna which costs approximately three dollars. There are additional kits which come with the antenna and card already connected for similar prices. VISION intends to equip the Jetson Nano with a Wi-FI and Bluetooth adapter.

5.8 Processor Selection

The Jetson Nano 4GB Development Kit is the desired processor for this project. The Nano is a high-performance embedded computer equipped with a powerful GPU that can be used for machine learning, artificial intelligence, computer vision, and other computationally complex tasks. The Jetson Nano is more than capable of performing all of the benchmark machine learning frameworks. The Raspberry Pi and Coral Dev boards could perform some of the benchmark tests, but there were many tests that the boards could not support. The Nano's ability to support a variety of machine learning tasks is what makes the board so desirable.

There are benchmarks where the Coral Dev board does outperform the Jetson Nano. However, the large number of benchmarks that the Coral Dev board could not complete is worrisome. The Coral Dev board was purpose-built for TensorFlow Lite and it appears that not even the standard TensorFlow framework can always be implemented on the board. VISION does not intend to use TensorFlow Lite, so it would be risky trying to use the Coral Dev board to run software that it was not designed for. Although the benchmark tasks were mainly related to real-time video processing, the results display how versatile of a device the Nano is.

Compared to the other boards, the Jetson Nano does lack Wi-Fi and Bluetooth capability. Although an ethernet connection can be used in place of Wi-Fi, there is a large portion of the project that relies upon Bluetooth for communication. There are numerous adapters available on the market that can be added to the Nano to provide both Wi-Fi and Bluetooth connectivity. The Edimax N150 adapter is a 2-in-1 Wi-Fi and Bluetooth 4.0 adapter that plugs directly into one of the Nano's USB ports. This adaptor is relatively inexpensive and significantly increases the usability of the Nano.

Furthermore, the available port selection on the Jetson Nano is more than sufficient to support all of the peripheral devices needed by VISION. The Jetson Nano has a USB-C 3.0 port , a USB-C 2.0 port , two USB 2.0 ports , a USB 3.0 port , HDMI port, ethernet port, and 40 GPIO pins. With the addition of the Wi-Fi and Bluetooth 4.0 adaptor, the Jetson Nano will have two forms of wireless connectivity.

To ensure that the Jetson Nano can support all of the peripheral devices needed, figure 5.8 shows the tentative connection diagram for the Jetson Nano. The Jetson Nano is the central processing unit for VISION and will coordinate communication with all of the other devices.

A significant amount of communication will be done using wired connections. The USB-C 3.0 port will be used to power the Jetson Nano from a wall power outlet. The USB 3.0 port will be used to communicate with the web camera for the computer vision system. The HDMI port will be used to display information on the monitor for spectators. Some of the GPIO pins will be used to establish an SPI connection to an ESP for user guidance. The ESP for user guidance will take in guidance information from the Jetson Nano and handle guiding the user to the necessary location. The implementation of this system and how it interacts with the speaker array is abstracted away from the Jetson Nano.

VISION will also have to implement wireless communication to work efficiently. The Jetson Nano will use a Bluetooth connection to an ESP that will be used for user localization. The Jetson Nano will inform the ESP on when to begin user localization and relay the user's location to the user guidance system. Similarly to the user guidance subsystem, the implementation details of the user localization subsystem are hidden from the Jetson Nano. The user localization ESP will also work as a server for the Bluetooth beacons that determine the user location. There will also be a second Bluetooth connection established by the Jetson Nano that will communicate with the other team's central processing unit (a Raspberry Pi). This Bluetooth connection is how the two teams will communicate shot information.

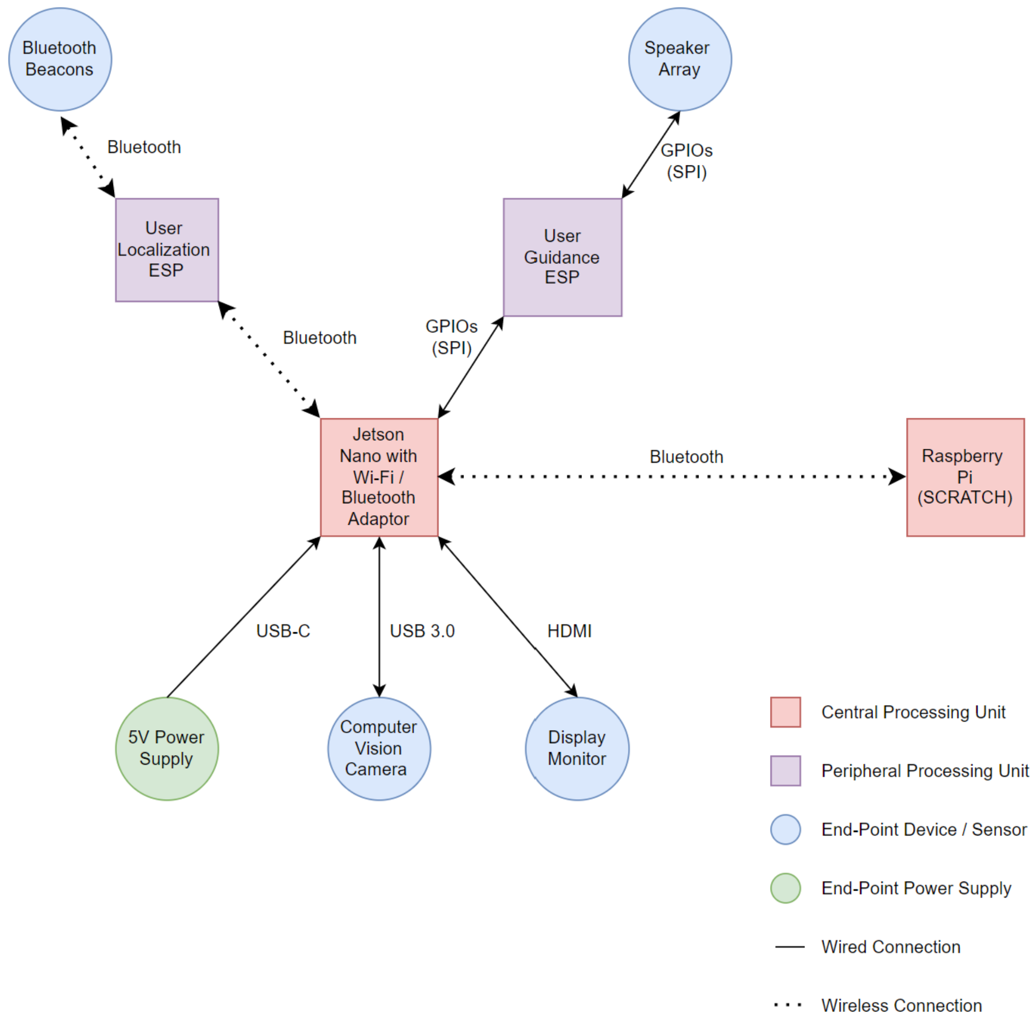


Figure 5.9 Jetson Nano Device Connections

6. SYSTEM SOFTWARE DESIGN

6.1 Pool AI

Extensive research for the shot selection algorithm has been completed in section 3.2.1. With many possible implementations to choose from it is important to first clarify the system requirements.

- Input: List of current table state, this is the (x,y) location of every ball, along with the classification of every ball.
- Output: The force and angle to hit the cue ball

Summary of Requirements:

- Algorithm produces output in under 20 seconds
- Algorithm produces shots in which the end of a 3 to 4 foot pool cue will not intersect with the dimensions of the table
- Ensure that 1 foot from the cue to the shot angle does not intersect with any balls

The algorithm must be quick enough as to not impede the game flow. If an algorithm takes more than 20 seconds, we will cut down on its accuracy and how many moves ahead it is planning. The user will likely not be hitting every ball in, so branching into the future too far is a waste of computational power. The algorithm must also make the correct decision in a very simple situation, prioritizing simple shot suggestions over more complex shots, even if advantageous.

Using an existing shot selection algorithm out of the box is currently not an option. Many are slow and connected to GUIs. They also lack the constraints of a real table, and will suggest shots which are not physically possible.

Timing Considerations: The existing shot selection algorithms will be stripped of their GUI for production mode, possibly increasing performance. The search and heuristic based algorithms have built in physics engines which are required, these cannot be offloaded and decrease performance. The branching factors of the algorithms can be diminished to a smaller amount. While the algorithms are built to win on a single turn, we do not expect nor need this level of accuracy. Reducing branching will dramatically speed up performance.

Realistic space considerations: The algorithm must give the player a shot which is reachable. For example,consider the shot shown in the top of figure 6.1. Even though this would be the best shot, there is no way the player could reach this. The better shot alternative would be something such as this the shot shown in the bottom of figure 6.1.

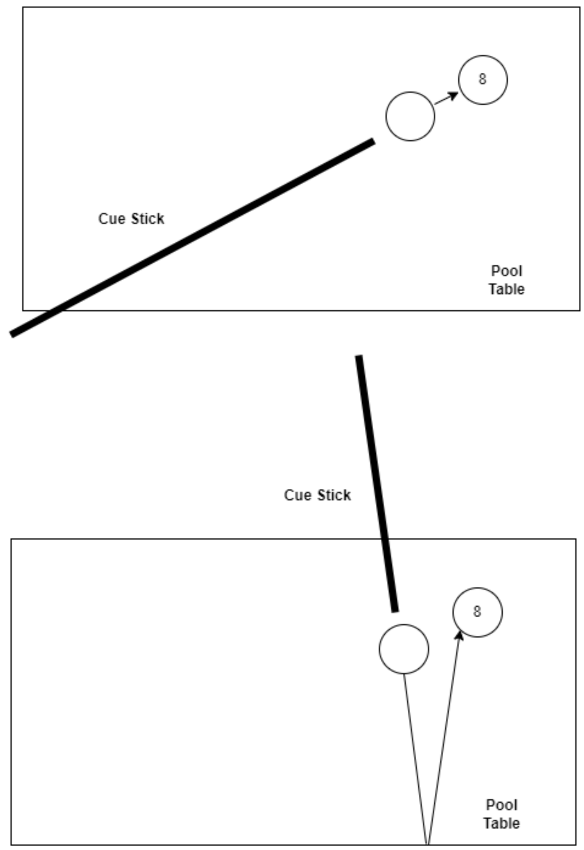


Figure 6.1: Example of Reachable Shot Issue

The main problem will be how to design the algorithm so that it only considers realistic shots. An additional algorithm will have to be made to ensure the length the pool cue is from the pool table wall is not too far. User testing must be conducted to get the exact length in question, but the algorithm will follow these steps. The algorithm must also take into account the width of the user's body. On one side of the table, the user's body will be in the way, on the other side, the user will have much more mobility.

<u>*Algorithmic Process:*</u> Below is the general outline of an algorithm with relevant parameters defined.

Max Extension= maximum distance the cue stick can be over the table
Current Extension = total distance the stick is over the table
User Width = the average space the user takes up
Shot Angle = Angle the cue stick will hit the cue ball at
Cue Ball Coordinates = The center of the cue ball given in x,y
Cue Ball Radius =
X Min = This is the left side of the pool table and represented by 0
Y Min = This is the top of the pool table and represented by 0
X Max = This is the right side of the pool table
Y Max = This is the bottom of the pool table

1. The shot selection algorithm produces a possible shot angle

2. Following the proposed shot angle, extend a line from the edge of the cue ball to the edge of the pool table. This Distance will be the stickExtension.
   Finding this distance algorithmically is not as simple as extending out the line though.
    a. Determine the quadrant 1 through 4
    b. Create a small triangle inside of the pool ball, use the radius as the hypotenuse and the given angle, then use sin and cos for coming up with the x and y distance

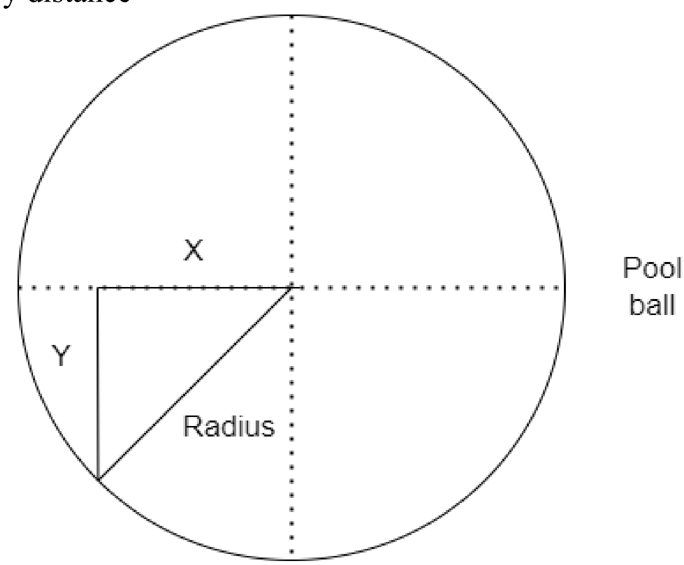


Figure 6.2 Shot Angle Projection

    c. Depending on the quadrant, you will find the minimum distance from the center of the pool ball to the corresponding x and y value for the side of the table. This will be called the minimum difference. You will also record the corresponding axis, x or y, this will be called the minimum difference axis.

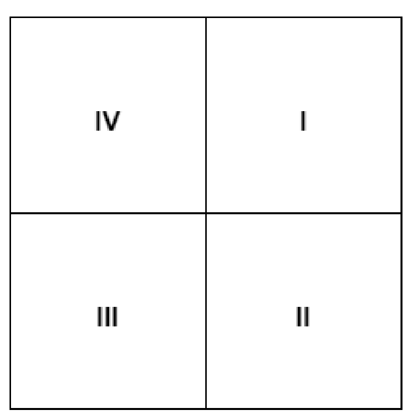


Figure 6.3: Shot Angle Quadrant

        i. Quadrant I: 0 for y, max for x
        ii. Quadrant II: max for y, max for x
        iii. Quadrant III: max for y, 0 for x
        iv. Quadrant IV: min for y, min for x

    d. Divide the minimum difference by the corresponding length on the minimum difference axis of the small triangle. This will give you the extension factor
    e. Multiply the radius of the pool ball by the extension factor and subtract one radius from it, this will give you the current extension

3. We will then check to see if stickExtension is greater than stickMax, if it is, the shot will be skipped.

4. Next we will check to see if the user's body is in the way of the shot. For this, we will extend a line the length of userWidth at a 90 degree angle and to the left of the stickExtension line. If this line does not intersect with the dimensions of the pool table, we will accept the shot. If it does intercept, we will continue to the next step.

5. We will now extend currentExtension to maxExtension beyond the pool table wall. From here we will once again extend a perpendicular line the length of userWidth to the left of the maxExtension line, if this line still intersects the table, we will skip the shot, otherwise the shot is deemed acceptable.

A separate algorithm which also falls into the category is room for the pool cue to move without the interference of another ball. This algorithm may seem simple but requires more advanced geometry. Algorithms like these are found in many 2D games and we will base our work off of the common algorithms. Raycasting is used in many games and a similar algorithm will be used to ensure that the shot does not intersect with other balls.

1. The cue ball position will be deconstructed into its x and y position
2. Create a unit vector
    a. unit vector x = cos(shot angle)
    b. unit vector y = sin(shot angle)
3. Loop through every ball on the table currently
    a. Take the ball_x and ball_y from the ball
    b. Create a vector from the origin to the ball
        i. Origin_to_ball_vector = (origin x - ball x, origin y - ball y)
    c. Get the magnitude of the bal vector
        i. Magnitude ball vector $= \sqrt{(origin\ to\ ball\ vector\ x)^2 - (origin\ to\ ball\ vector\ y)^2}$
    d. Compute the intersection
        i. Intersection = unit vector x * origin to ball vector x + unit vector y * origin to ball vector y
    e. Calculate interaction length

i. Intersection length
$$= \sqrt{(magnitude\ ball\ vector)^2 - (intersection)^2}$$

f. If intersection is greater than the radius of the ball then the raycast intersects

Modifying of “PoolGenius” : The open source project we have decided to modify for our purposes was described in section 3.2.1. While there are several issues with software, we have decided that there are several factors making a high accuracy simulation and shot selection algorithm unneeded. The overall uncertainty of the player being able to match the force and the angle perfectly make strategic planning almost useless. It also makes the need for perfect physical simulations of 3D objects along with friction and other resistive properties not needed. What is needed is believable simulation of collisions which produce shot selections which a real player would see as logical. Pool Genius already has a collision system and AI, we will be making the following modifications for our project needs.

- Table state changes : Must be able to set the simulation table state to the real table state after every shot. This can be accomplished by changing the program to be fed the current table state and then producing a shot before closing
- Implement above algorithm to see if the shot is reachable by the player
- Implement above algorithm to ensure the pool cue is not be blocked

Below is a UML class diagram describing the design plan for integrating our constraints with the PoolGenius software. This UML diagram focuses on the parts our team will be implementing in conjunction with the simulation system used. I have not added all classes and functions due to the large nature of the software. I have instead focused on adding enough classes to enable a basic understanding of the PoolGenius software. The RealisticAI class inherits from the base PoolAI class in order to communicate with the existing simulations run by another physics software known as Box2D. The drawable class will have another function in order to draw a pool cue, this will allow for our GUI to better show the desired shot angle. There are two functions which will be added to the software, one is test_mode which allows for the GUI to be active and the other is production_mode which will run more efficiently without the GUI overhead. The test_mode function will also allow for results to be verified in an easier fashion.

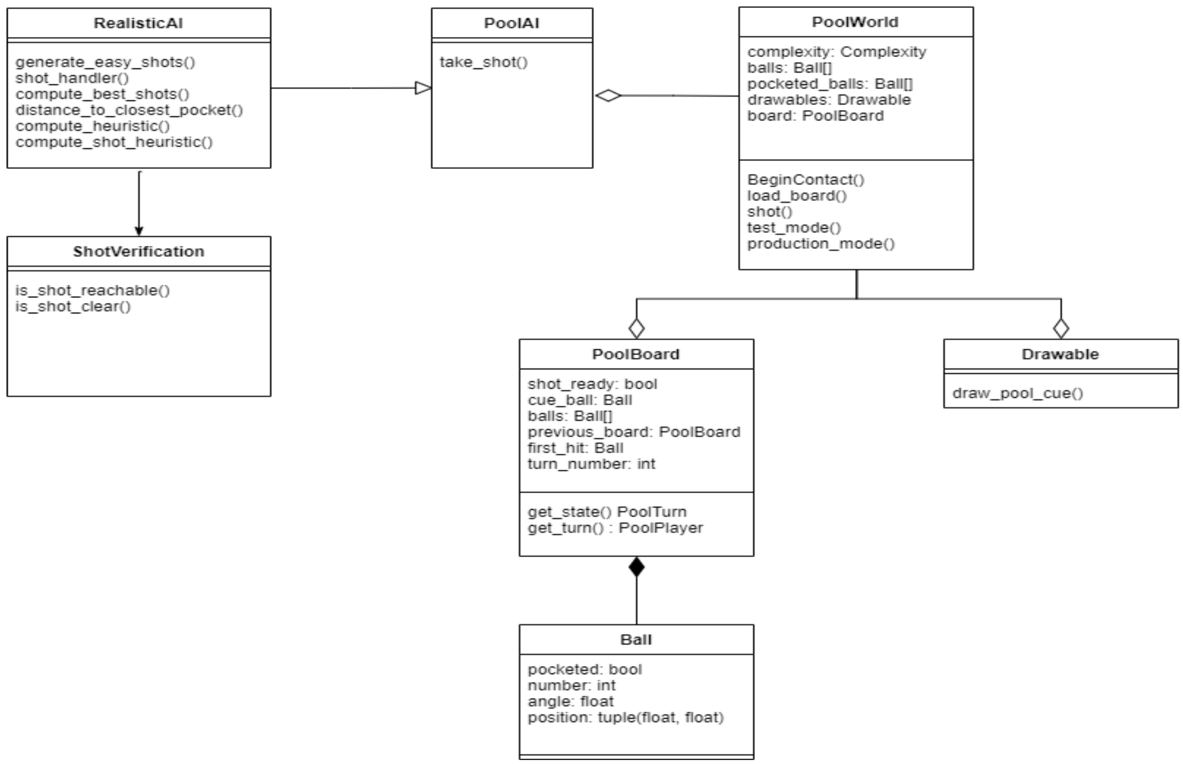


Figure 6.4: High-Level Overview of Shot Selection System

6.2 Computer Vision System Software Design

The system must be able to isolate the billiards table from the background, identify the billiard balls and their positions, and be able to distinguish the cue ball from the other billiard balls. Section 3.2.2 outlines some of the relevant computer vision algorithms, available in OpenCV, that can be utilized to reach the computer vision goals. This section describes how the computer vision system will be designed and what algorithms will be used.

Before discussing the specific algorithms chosen, it is important to discuss the inputs and outputs of the computer vision system and how the system will interface with the rest of the project. The initial input to the computer vision system, and the entire project, is an image of the current state of the billiards table. This image will be processed through a variety of algorithms and will output a CSV file containing elements and their relative locations. This file will then be used by the shot selection system to determine the best shot to take. The elements in the output file of the computer vision system will contain the relative location of the six pockets of the billiards table, the relative location of the billiard balls, and flags to differentiate between the billiard balls.

The input image for the computer vision system will be run through multiple separate algorithms to extract different information from the image. It is important to maintain the input image so that the same input can be used for all of the algorithms. For any algorithm that permanently modifies an image, a copy of the original input should be supplied rather than the original image. The original image must also be preserved so that it can be used by the output system when showing the user the best shot options.

The locations in the output file need to be relative locations rather than absolute locations. Relative locations refer to the distance, in pixels, from some reference point for a selected feature of interest. Absolute locations refer to the raw pixel location in the input image. Due to the input image including some of the unwanted background, all of the pixel locations that are found will need to be localized to a point of reference. The selected point of reference will be the top left corner of the playable area of the billiards table. This reference point is used to stay consistent with the coordinate system used by OpenCV and will also represent the location of the top left pocket.

For all of the billiard balls found by the computer vision algorithms, their relative locations need to be included in the output file. Additionally, a flag will also need to be included with each billiard ball entry to specify if the billiard ball is the cue ball, the black ball, or a game ball. The cue ball and black ball have more significance than the other billiard balls for many games, so these balls must be differentiated from all of the other billiard balls in play. For this project, a game ball is any billiard ball on the table that is not the cue ball or the eight ball. Figure 6.4 summarizes how the computer vision system will interface with the other subsystems of VISION.

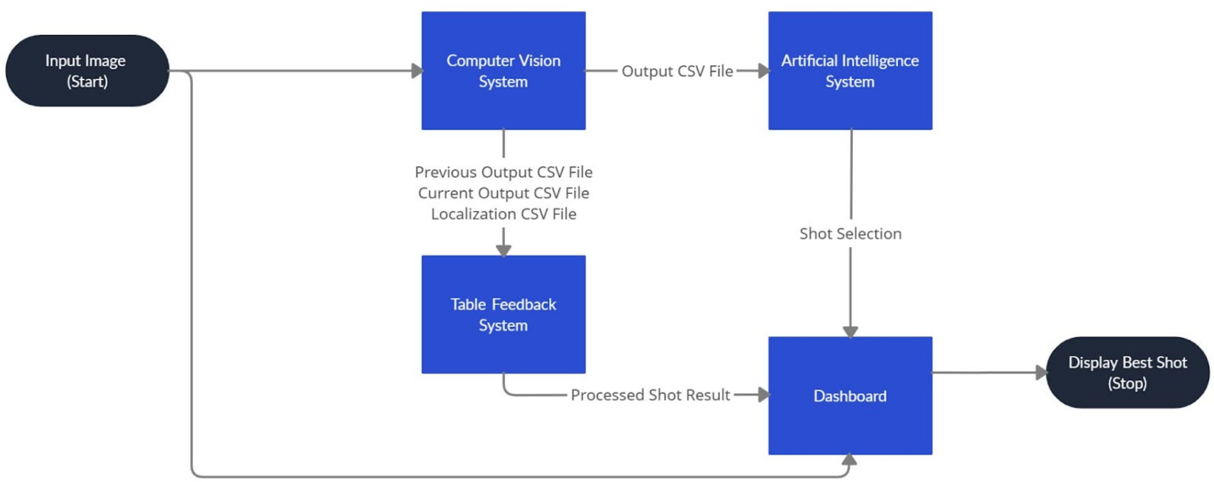


Figure 6.5: High-Level Overview of Computer Vision System

The diagram above summarizes how the computer vision system will interface with the other systems in the project. The initial input to the computer vision system is an image of the current state of the billiards table. The computer vision system will generate the output CSV file containing the localized coordinates of the pockets and billiard balls. The computer vision system will also create the localization CSV file containing the information needed to convert localized coordinates to the actual coordinates. The artificial intelligence system will use the output file to determine the best shot to take based on the current state of the table. The table feedback system will determine the outcome of the previous shot. The feedback system will use the localization file, previous table state, and current table state to determine which balls are no longer present on the table. The optimal shot, the shot results, and the input image will be used by the output display system to produce a visual of the shot for the user and spectators to view.

Billiard Table Isolation: The billiard table isolation portion of the computer vision system refers to being able to extract the playable area of the table from the input image. For this project, the playable area refers to the region of the billiards table where the billiard balls can be. This region is the nearly rectangular region of the table that is recessed from the borders of the table. Isolation is needed to localize the billiard balls to a reference point, verify that the contours found in the image are in the playable region, and determine the location of the pockets.

To isolate the playable region, the Douglas-Peucker algorithm will be used. This algorithm can be used to approximate the nearly rectangular region of the billiards table. The region of interest will be the rectangle formed by the interior table edges and the pockets of the table. This algorithm was chosen because it can approximate a contour with many edges into a much simpler contour with only a few edges. There are simpler algorithms that can identify the rectangular contours in an image, but they may not correctly identify the region because the corners are circular.

To localize the billiard balls in the image, a reference point needs to be chosen to localize the balls to. The upper left corner of the contour found by the Douglas-Peucker algorithm will be used as the reference point. As mentioned previously, this reference point is chosen to align with the coordinate system used by OpenCV. To localize the billiard ball coordinates to this point, simple arithmetic is needed.

The reference point, p, will have some positive, non-zero coordinates (x_0 , y_0). The reference point coordinates must be non-zero because the reference point will not be the upper left corner of the input image. If the reference point is assumed to be the new origin and denoted p^* with coordinates $(0 , 0)$. All of the billiard balls can be localized to the reference point p^* by subtracting (x_0 , y_0) from their coordinates. This transformation will ensure that all billiard ball locations are positive, non-zero values because no billiard balls can be above or to the left of the reference point. This claim can be made because any region above or to the left of the reference point is not in the playable region of the billiards table.

The localization of the billiard balls to a reference point can easily be reversed by adding the offset values, (x_0 , y_0), back to every billiard ball. The reversal of the coordinate system back to the true pixel values will be useful if any features need to be drawn on the input image. For example, when displaying the best shot to take it may be necessary to draw lines pointing from a billiard ball to a target ball to a pocket. For these lines to be drawn properly, the true pixel values, rather than the localized values, of the billiard balls need to be used. The localized values on the input image should only be used by the shot selection algorithm. To ensure that the original coordinates can be recovered, the offset values should be stored for the duration of the program execution.

Once the playable region has been discovered, it will be possible to determine if the contours discovered in later portions of the image processing are in the playable region. The borders of the rectangular contour found by the algorithm will have a minimum and maximum x-coordinate and y-coordinate. These minimum and maximum values can be used to ensure that any contour discovered in the image lies within the playable region of the table. If any object is discovered outside of the minimum and maximum coordinates, it can be discarded.

The rectangular contour outlining the playable region of the table can also be used to find the locations of all of the six pockets. Once the coordinates have all been localized, the upper left pocket will be at $(0 , 0)$, the upper right pocket will be at $(x_{max} , 0)$, the lower left pocket will be at $(0 , y_{max})$ and the lower right pocket will be at (x_{max} , y_{max}). The middle pockets can be computed by finding the midpoint between the two adjacent pockets. The top middle pocket will be located at $(\frac{1}{2}x_{max} , 0)$ and the bottom middle pocket will be located at $(\frac{1}{2}x_{max} , y_{max})$. Defining the pocket conventions this way means that the locations of the pockets only depend on the four corner values of the rectangular contour found by the algorithm.

The final output of this process is two CSV files. The first CSV file is the true output file that will contain the locations of the six pockets. This is the file that will ultimately be the output of the computer vision subsystem. The other file is an intermediate localization file only to be used within the computer vision system. This file will contain the offset value used to localize the pockets and later be used to localize the billiard balls. This intermediate file will also contain the minimum and maximum coordinates that define the playable region of the table.

Finding the Billiard Balls: To find all of the billiard balls in the input image, the Hough Circle Transform will be used. This algorithm was chosen because it is specifically tailored toward finding all of the circles in an image. The algorithm allows for the parameters to be modified as needed to only detect circles of a certain radius. This characteristic is useful because all of the billiard balls are of the same size. Once the expected radius of the billiard balls has been determined, the algorithm can enforce these restrictions on the circles found to ensure that only billiard balls are discovered.

Additionally, this algorithm was chosen for its ability to detect touching circles and partial edges of circles. The algorithm traverses the discovered edges in an image and looks for points of intersection, and assigns points to these values. For this reason, two touching billiard balls can still form two distinct radii which allows the algorithm to detect both billiard balls. This trait of the algorithm is especially appealing because other algorithms are sensitive to objects being too close together. This algorithm is also able to detect circles from partial edges. Even if there is only a portion of a circular edge present, this algorithm is still able to traverse the edge and identify that the edge represents a circular contour. This behavior of the algorithm is ideal for situations when the lighting is not optimal and there are shadows or unclear edges in the input image. The robustness of this algorithm is another reason why it was selected for this project.

Once the Hough Circle Transform has been run on the image, it will return a list of discovered circles. Initially, there will be no restrictions on the radius of circles returned so that the expected radius of the billiard balls can be determined. This testing will occur in various lighting conditions and with various numbers of balls on the table. Once a reliable minimum and maximum radius have been discovered, these parameters can be implemented into the algorithm. Including the minimum and maximum radius will allow for the algorithm to automatically exclude any contour that is too big or too small.

The final step of this algorithm is to write the discovered billiard balls and their locations to the output file. This output file will be the same output file from the table isolation stage and will already include the locations of the six pockets. A separate file from the table isolation stage containing the localization value and the minimum and maximum allowable coordinates for the billiard balls will also be needed.

The list of all discovered circles will be iterated over and all of the coordinates will be localized to the reference point. The locations of the contours will be checked for being in the playable region. If the coordinates of the contour fall within the playable region, the location is added to the output file that contains the coordinates of the pockets. If the coordinates of the contour are not in the playable region, that contour is ignored. All of the locations added to the output file will have the keyword *game_ball* added to them to serve as an identifier for future processing. The necessary modifications needed for identifying the cue ball and the eight ball are discussed in the next section. The output of this part of the computer vision system is the output file with all of the discovered billiard balls and their localized locations appended.

<u>*Detecting the Cue Ball and the Eight Ball:*</u> While the previous section outlines how to detect all of the billiard balls in the input image, special consideration is needed for the cue ball and the eight ball. In nearly all billiard games, these balls have more significance than other game balls. For this project, the cue ball represents the only ball that the player can hit directly with the pool stick (a simplification of 8 ball pool). The eight ball is the final ball a player must hit to win the game. Due to their significance, the cue ball and the eight ball need to be able to be distinguished from all of the other billiard balls that are detected. Once these balls have been detected, they will be given a special keyword in the output file. Rather than being called a *game_ball*, the cue ball will be given the *cue_ball* tag and the eight ball will be given the *eight_ball* tag.

The ideal way to detect these balls is to make a small addition to the previous section. The previous section outlines how to find and filter all of the circular contours in an image using the Hough Circle Transform. An additional step can be added to this process to check if the color of the discovered contour is white or black, indicating that the cue ball or the eight ball has been found. Although the transform requires a binary image as input, the locations of the contours that are found can also be applied to a color version of the same input. This allows the color of the discovered contours to be checked before adding these locations to the output file.

The RGB values of the discovered contours can be compared with a predefined threshold value. A perfectly white RGB pixel will have the values of [255, 255, 255] for the red, green, and blue color channels, and a perfectly black RGB pixel will have the values of [0,0,0]. A lower bound can be experimentally determined such that the cue ball can be reliably identified and a similar approach can be used for an upper bound on the eight ball. As long as a contour's color channels are within the threshold range, that contour can be considered the cue ball or eight ball respectively. It will be important to determine threshold values that do not provide any false positives when iterating through the contours. This color check can be implemented right before a billiard ball's location is added to the text file. If the contour represents the cue ball or eight ball the location will be given the *cue_ball* or *eight_ball* tag. All of the other contours will be given the *game_ball* tag. Figure 6.5 summarizes the computer vision system.

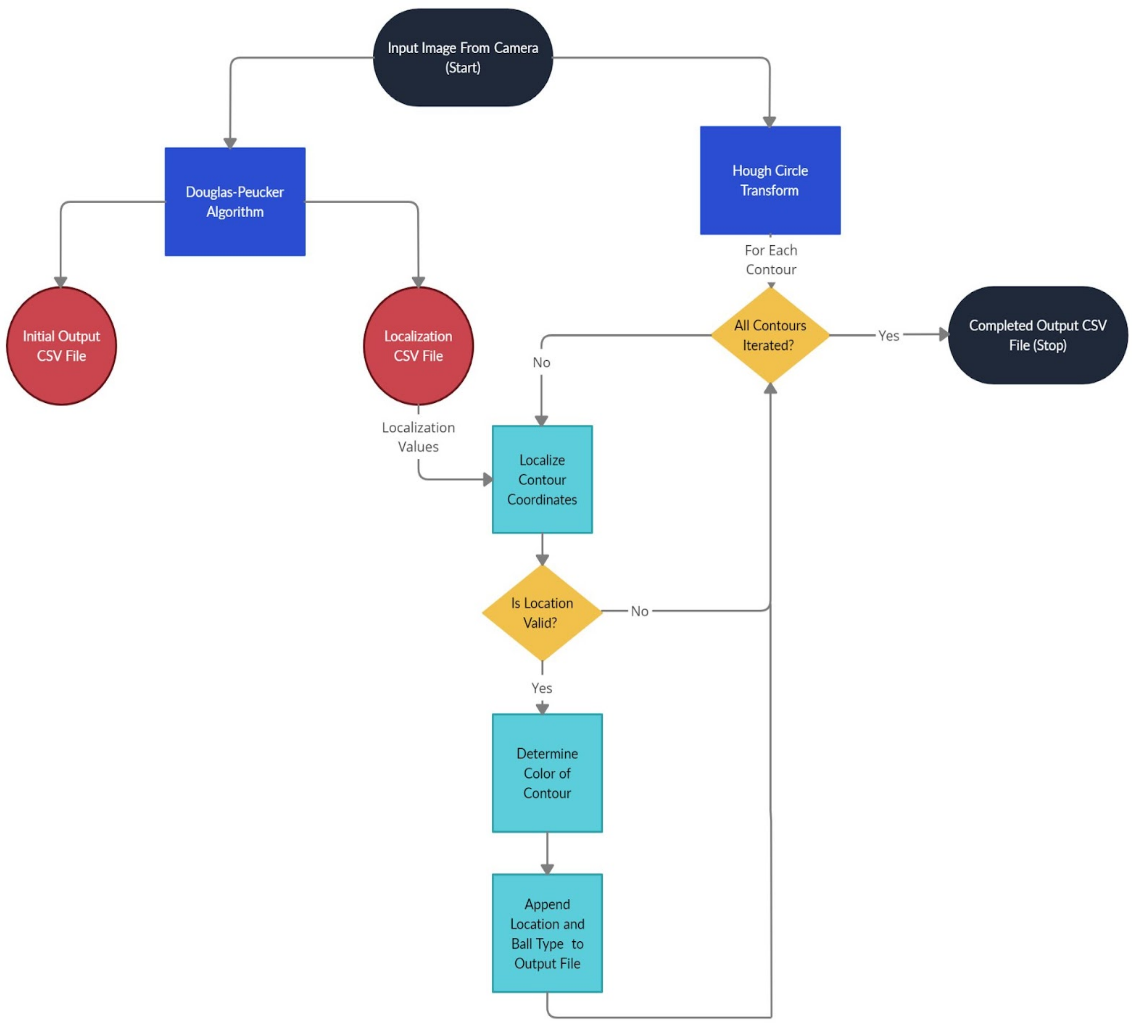


Figure 6.6: Computer Vision Implementation

The diagram above summarizes the computer vision system. The input to the system will be an image of the current state of the billiard table. The Douglas-Peucker algorithm will be run to isolate the table from the image background. The output of this algorithm will be the output CSV file and the localization CSV file. At this point, the output CSV file will contain the localized coordinates of the six pockets. The localization file will contain the offset values needed for localization and the minimum and maximum x-coordinates and y-coordinates.

Once the Douglas-Peucker algorithm is completed, the Hough Circle Transform will be run. The Algorithm will output a list of contours that represents the billiard balls. The diagram does not explicitly show the size of the contours being checked. The size check is omitted because the Hough Circle function in OpenCV takes the minimum and maximum radius as parameters and automatically excludes any contour that is not within the acceptable range. The list of contours will then be iterated over. The contour location is verified to be within the playable region. If a contour is not in the playable region, it is excluded. The contour's color is then checked and the contour's location and associated keyword are then appended to the output file. Once all of the contours have been checked, the output file is ready to be used by other systems in the project.

6.3 Navigation Algorithm Design

VISION's core goal is to navigate an impaired user to a desired location. The following is the design in place to make this primary goal a reality.

6.3.1 Localization Algorithm Design

ESP32: The biggest part of the software design for our localization system will be on the ESP32. As described in the hardware section, we plan on using BLE advertising packets received from three Bluecharm beacons and Wi-Fi as our main tools to determine the position of the user as they move around the table. We will use the Arduino IDE for our programming purposes and install on it the ESP32 board related to our ESP32 Devkit V1. This will allow us to make use of different header files written in C++ as well as their associated source code to make use of both the BLE and the Wi-Fi capabilities of the ESP32. On the BLE side, the main header files or namespaces that we will use are the following:

- *BLEDevice.h:* Allows initialization of BLE functions on the ESP32
- *BLEScan.h:* Allows scanning for other Bluetooth devices in the vicinity of the ESP32
- *BLEAdvertisedDevice.h:* Allows reading of specific features of the scanned Bluetooth devices including but not limited to their UUID, RSSI, Transmission power, address, names if applicable.

The goal of the code for the BLE section will be to initialize the device using properly chosen and tested settings during the setup phase and continuously scan for devices in the close vicinity of our device. Once scanned, the devices will be differentiated by their address which we will need to access from the Bluecharm beacons. To do this, VISION will make use of the KBeacon app recommended by the company producing the beacons. This step will only be done once as it is not part of the overall process of our project. The KBeacon app basically allows users using a phone to scan the Bluecharm beacons and access different settings such as the beacon's address, modifiable advertising interval, beacon type, transmission Tx and measure power, beacon main UUID, major and minor UUID among others. The main discernable settings we will need to leverage are the address to determine which of the scanned bluetooth devices are one of our three Bluecharm beacons as well as doing testing with the advertising interval setting. Once any of the three beacons has been detected and properly identified, VISION will either store the RSSI values of the detected beacon temporary and release prior stored values to save on memory space within the device or continuously send out one RSSI reading at a time.

These RSSI values will be sent out via Wi-Fi. The headers or namespaces we will make use of are:

- *Wifi.h:* Allows setting up and connecting to a Wi-Fi network in the vicinity
- *WifiUdp.h:* Allows use of UDP as communication protocol for Wi-Fi. This header file allows data, mainly numeric data, to be sent out as packets to a specific address and port. The address will corresponding to the IP address of the main computer and the port is a chosen number used for differentiation purposes.

During the setup phase, the ESP will connect to a predefined WiFi in the area in which our overall project setup will be located. Then the ESP will continuously loop through and send out the RSSI readings that have been updated either in real time every time the loop runs sending out one value at a time or after a specific amount of time which will allow a chunk of values to be read at a time and sent to our main computer. Testing will need to be done to see which one of the two options will be better both in terms of latency and for proper use by the guidance system. The second option is advantageous in that the sent out RSSI values can be averaged out for the specific period of time and will allow a more accurate position to be determined for the user's location. However, this does not work ideally if the user is in constant motion as the RSSI values will be rapidly changing. The first option takes in mind the fact that the user's position is constantly changing by computing a position every time but depending on latency in the code itself, determining the right position of the user at any time might still be overshadowed if the user moves faster than the system can compute their position. All of these factors will be taken into consideration and the most user friendly, yet computationally reasonable section will be implemented.

<u>*Jetson Nano:*</u> The main computer is the Jetson Nano where a Python file will continuously run to read the RSSI values sent via Wi-Fi. Python is chosen because of its flexibility through available modules for a wide range of applications, can be ran directly and in a single file from the shell or command line and can easily interact with other portions of the code. Bluetooth sockets will be used to communicate between the Jetson

Nano and ESP32. The socket will bind to a specific host and the same port we specified on the ESP32 and continuously receive data via Bluetooth. Data preprocessing will be needed prior to sending data in order to differentiate between the different beacons depending on how they are being sent from the ESP32.

The code on the Jetson Nano will then compute the current distance between each beacon and the ESP32 from the RSSI values using the trilateration formula discussed previously. This part of the code will mainly be mathematical calculations in which the end x and y position of the user will be computed from the individual distances from each beacon to the user.

The computed value can then be sent and used by the guidance system either every time it is being called, or the real time position can be continuously calculated and stored in a CSV file. The latest positional entry in the CSV file can then be read by a separate code used for guidance of the user. Each of the decisions and uncertainties listed in this design section will be taken into account, tested and the best solution for every subsystem implemented at the end.

6.3.2 Guidance System

Guidance of the user is dependent on three passed parameters: current location of the user, the destination determined by the shot selection algorithm, and the layout of the table. As these are passed, the user and destination will be placed within the defined array encompassing both the table and traversable perimeter. As these are placed, a calculation will be made on which of the binary routes is shortest and then deploy a route to be taken. As this route is determined, a speaker will be turned on and utilize a digitally generated PWM signal producing a desired output for the user. As this process occurs, the software will continually update its input parameters until destination and location are equal to one another. This means the user is properly located at the appropriate shooting position on the table.

Once arriving at the desired location, a calculation will be made on which speaker has the best correlation with the line of attack of the cue stick. Once this is determined, the speaker will play the same generated output previously described for a 10 second period. Upon completion, the software will wait for an input from the user control system to signal a shot has occurred. This begins the process over again with the computer vision algorithm as there is no active feedback loop to correct for positioning errors outside of the SCRATCH system.

6.4 Web Interface

The web interface will be a dashboard built in to provide continuous, real time, and updated information to the user and people around the pool table about the user's performance and state of the game as it progresses. Since the web application will have to rely on working with data and displaying the data on a web page, VISION aims to find tools that would allow a simple way to display and update the data. This section examines different options and then describes how to use the better option in the software design.

R vs Python: Two most common programming languages for data analysis and statistical computation are R and Plotly. For this reason, a comparison of these two programming languages is presented before diving into which modules offered by either languages would serve the best purpose. Table 6.1 describes the comparison between the two options.

Comparison Criteria	R	Python
General overview	Open source interpreted (runs on command line) programming language for statistical computation and graphics.	Open source high level, interpreted general purpose programming language. Modules on Python such as Pandas, Scikit, SciPy, Seaborn, ggplot2, Matplotlib on Python allow to cover some of the statistical, data modeling, and data analysis functions that are inherent to R
Syntax	More complex to gain expertise in	Easy to read syntax makes learning curve much more linear and simpler
Libraries and packages flexibility	R modules are easier to use and more powerful overall. However, for our current functionality, we don't expect to require any intensive computations	Python modules are more complex to understand and less powerful. This learning curve, however, is not comparable to that of learning R from scratch in our case.
Data collection	R is mainly limited to CSV, Excel and txt files.	Python allows you to use data from CSV, txt, or other data formats. Also allows importing data from SQL databases into the python code. You can also request data in JSON format through web requests and use said data in your web page or web design
Memory usage and speed	R consumes more memory since the objects created are stored in physical memory. As time goes on, this will lead to slower overall functionality. R is also naturally slower than Python, taking more time to return outputs regardless of the code.	Python is relatively slower than other programming languages since it's an interpreted language but is still faster than R. It does also consume significant memory space so special care will have to be taken by deleting unused data as time progresses or clearing memories as needed.

Table 6.1: Comparison between R and Python

Looking into the options on Python for designing our web based dashboard. Two standouts that require more attention to determine which one to use. The two options are Dash and Streamlit. Both Streamlit and Dash are full dashboarding solutions for Python based data analytics and rely on Tornado and Flask respectively to deploy the dashboard on local or web based servers.

Streamlit is more focused on rapid prototyping providing a fully fleshed dashboard with as little code as possible. However it suffers from other drawbacks that should be considered. Dash on the other hand focuses on more robust production/enterprise dashboards, which may be overkill for our application, but allows much more flexibility and options for what can be done despite the more difficult programming barrier. In itself, a fully fleshed local web server app can be launched from both Python modules in a relatively small amount of code (as shown in appendix B).

Where dash_html_components and dash_core_components are other modules associated with dash that allows VISION to use most if not all of the available html elements from regular web page designs and construct different graphics respectively. In this case, the basic app layout would consist of an html Div element or grouping element under which the app would create a graph using dcc.Graph whose id is main-graph. The id value here works in the same way that ids (and classes) work in CSS.

Dash is chosen over Streamlit due to the reasons outlined below:

- CSS flexibility and aesthetics: Dash allows a user to fully customize the id and classes as mentioned below that correspond to different divisions or sections of our web page through a .css page in the same way that would be done for a regular web page. Also, just like regular web design, a user can take advantage of both Bootstrap themes and Bootstrap elements in order to fully customize a webpage to match specific color themes or modify the aesthetic of different elements such as buttons, dropdowns, depending on what will be needed in VISION's web page during the design phase. Hence, for users not fully familiar with CSS, they can use prepackaged CSS files such as the ones provided through bootstrap or other available files or snippets of files online to modify their web page.

- Similarity to HTML and CSS: For anyone with previous web page design experience, Dash is much more intuitive and easy to follow. The same perks that are known to work with regular webpage division are applicable when modifying a dashboard through a Python dash script. In addition to this, a user can modify the JavaScript elements of the webpage directly if needed. Dash_html_components introduced earlier contains a plethora of basic html features providing pure Python abstraction around HTML, CSS, and JavaScript. For users not fully familiar with HTML, only a few of those are actually needed to design a fully functional dashboard, which reduces the worry of having to learn HTML/CSS. In addition to that, being able to use HTML elements also allows a user to modify their styles directly bypassing CSS pages, or group similar elements into classes or ids which will in turn share the same CSS properties.

- Callbacks in dash: For fully interactive and constantly updating dashboards, Dash offers Python functions disguised as callbacks which basically modify the inner characteristics of different elements in a web page set as output due to other characteristics of elements set as inputs or states.

- Documentation and module integration: Streamlit allows us to integrate a lot more Python modules into our design such as OpenCV or TensorFlow when compared to Dash. Dash has the advantage for this project because a lot of the additional rendering from the computer vision side of our project would have been done beforehand and a simple image of the end result can be then added to the dashboard. Also Dash has an extremely broad community support, where many questions have been asked and answered before.

Considering the inner complexity of Dash, it is also worth dividing the dashboard coding process into distinct files, as described by the flow diagram (figure 6.6) below.

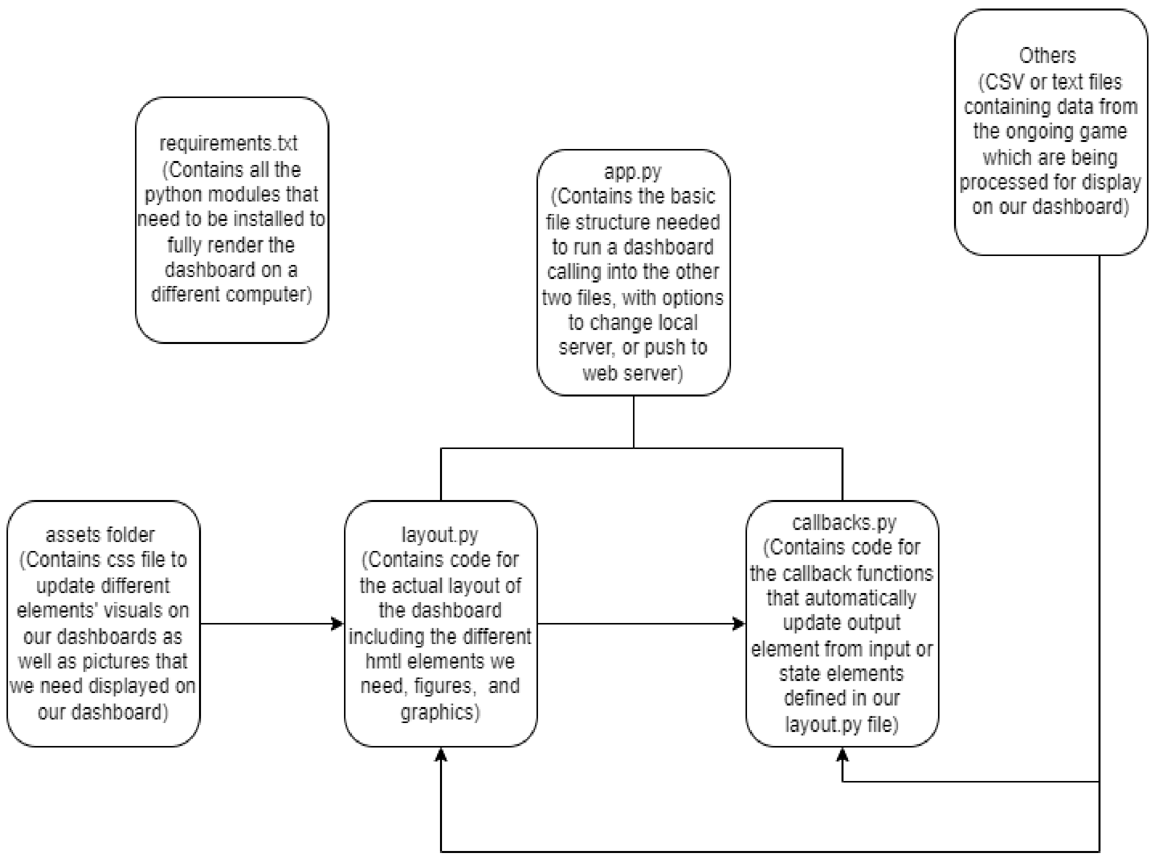


Figure 6.7: Senior Design Dashboard File/Folder Structure

Deploying the App: Dash is built on top of Flask and uses Flask as its web routing component on a local web server or on a server accessible to everyone on the same network. A dashboard app can either be created to run the localhost's IP address only with specified ports if they are available or can be configured to run on host 0.0.0.0 and then any specified port which will allow the app to be accessible to anyone on the network's IP address.

To make the app available directly online, Dash provides different options for deployment through Heroku. Only a Heroku account, Git, and a virtual environment on Linux where the Python modules needed for the app will reside.

An alternative to Heroku that could be easier to use is PythonAnywhere. PythonAnywhere works in a similar fashion to Heroku. The main difference is it does not require Git and uses uwsgi instead of gunicorn to populate the server. From a programming perspective, there is not a significant difference between the two server hosting softwares. Figure 6.7 shows the tentative layout of the dashboard for VISION.

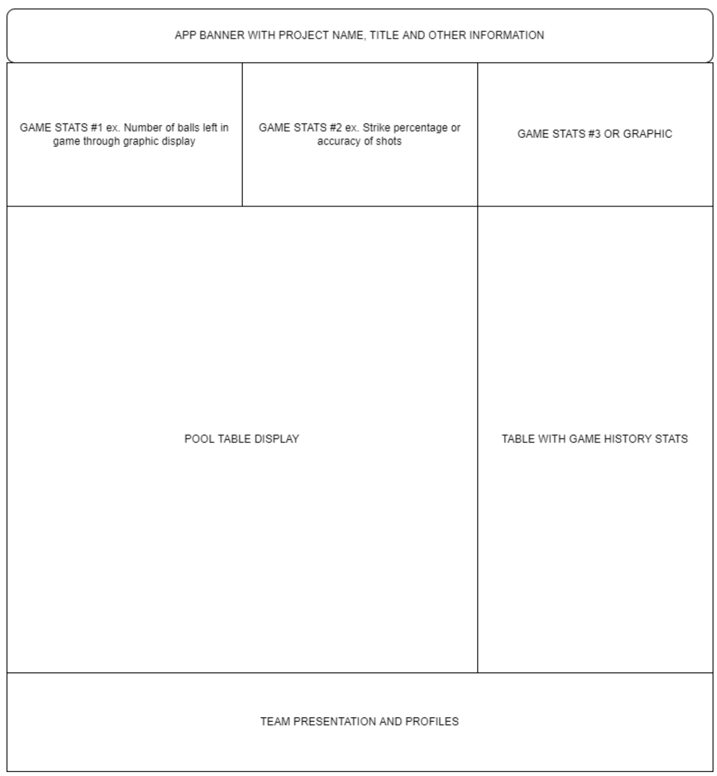


Figure 6.8: Dashboard Layout

6.4.1 Output Image Generation

Creating the Output Image: The display will include many important statistics and visuals for the user and spectators to view. Of the displayed outputs, the most significant image is the ideal shot for the user to take. The shot will be displayed by placing lines between billiard balls and the desired pocket on the input image. One line will connect the cue ball to the desired game ball and another line will connect the desired game ball to the desired pocket. To create this output image, artificial intelligence and computer vision subsystems will be needed. The computer vision system will provide the input image and localization CSV file and the artificial intelligence system will provide the optimal shot selection.

The display system does not perform any significant computations, but rather uses the output of the other systems to produce a visual aid. To draw the ideal shot path, only the targeted game ball and the desired pocket are needed from the artificial intelligence system. It should be noted that the preferred shot selection will be calculated using localized coordinates rather than absolute coordinates. For this reason, the coordinates of the shot selection will need to be transformed back to the absolute values before being used. A previous section describes the transform and inverse transform of coordinates in greater detail. Once the raw coordinates of the cue ball, the desired game ball and pocket have been calculated, OpenCV can be used to easily draw the ideal shot selection for the display.

7. SYSTEM FABRICATION

With the extensive physical and design footprint of the VISION apparatus, a fabrication plan is put forth for both PCB and the full system.

7.1 PCB Design

To properly integrate the circuitry components of VISION and satisfy a simplistic design for integration, several core components will be conjoined through a printed circuit board (PCB). The following section provides details on how the design will be conducted and the best practices to provide a functioning product. For the purposes of VISION, the PCB will be designed in EAGLE for its easily used interface in free usage as students at UCF. The majority of components that will be built into the PCB can be accessed using the EAGLE libraries, imported libraries from distributors such as Digikey and Mouser, and custom components when needed.

7.1.1 PCB Design Philosophy

The following outlines important practices in PCB design as outlined from Altium, one of the leading PCB development software companies. (Peterson)

Component Placement: Component placement is where PCB begins and can be fine tuned throughout the process of development. The goals for a well placed board should focus on ease in routing and limiting layer changes when possible. Several good practices to ensure a proper layout consist of prioritizing placing must-have components first and large processors/ICs in central locations, avoiding net crossing, placing all surface mount devices on one side of the board, and experimenting with different orientations of components. Following these steps and focusing on the largest and biggest hassle components first can limit headaches and improve design throughout the PCB design process.

Power Planes: Following the placement of components, the orientation of the power and ground planes is the next focus. Power and ground are placed on two internal layers, which can be a hindrance with only two layers. The ground plane ideally is on its own layer and is recommended as to not have to route ground traces on a board. Power is recommended to be implemented via common rails connected directed to the power source, but power planes can also be implemented if components do not get daisy chained and have wide enough traces implemented.

Routing: Determining the proper routes for connections between components can be an artform and is very up to the designers discretion. Ideally, short and direct routes are highly recommended. An important rule to follow is if all the traces on one side of the board flow in one direction (horizontal), the other side should flow all traces the opposite direction (vertical) to restrict emf disruption along traces. This is very important in two layer designs, and should alternate between layers in multi-layered board designs. Certain special case designs will require added practices to account for specialized component characteristics. Additionally, determining the proper width for traces can be a complex process, but can be determined by analyzing the manufacturability, current consumption, and impedance that will be seen through the design.

Component Grouping: Guidelines on grouping and separation can be valuable to ensure easy routing, prevention of electrical interference, and thermal management. At the heart of component grouping is placing items that are in a circuit together, especially if they do not interact with other portions of the board. Separating analog and digital components is a very important step in grouping, and can prevent commonly introduced interference. If these grouping practices are followed, the design becomes an exercise in placing groups rather than individual components. An important note in the grouping process is the separation of high powered components, as close proximity can lead to thermal issues.

7.1.2 PCB Design

The components of VISION included within the project's PCB are centered around the guidance output system and the user control interface. This encompasses a connection to the Jetson Nano, outputs to each speaker, regulators for both voltage and signal output control, a demultiplexer for signal selection, and push buttons for the control interface. Included in the PCB are the following major subsystems and components:

- Connection to Jetson Nano
- ESP32 Chip
- Switching Regulator
- Audio Amplifier
- 12 Speaker Outputs
- Demultiplexer (CD74HC4067)
- Four Push Buttons

Figure 7.1 shows a block diagram of the systems included in the PCB design.

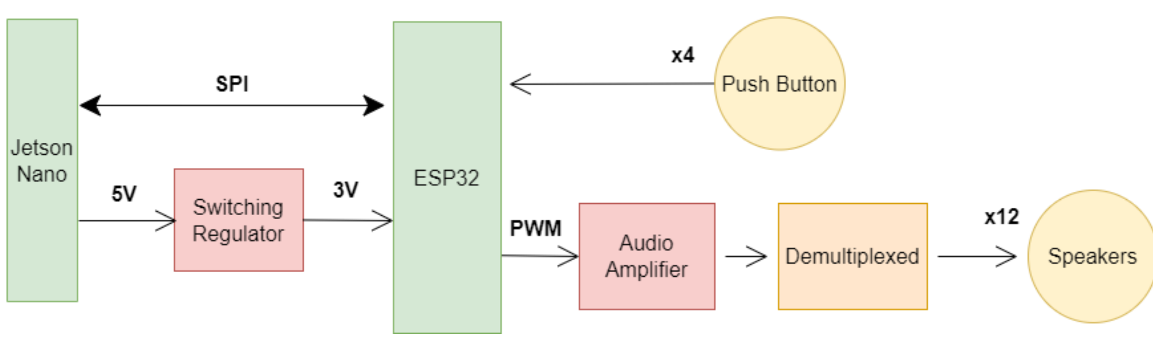


Figure 7.1 PCB Design Block Diagram

8. SYSTEM TESTING PLAN

The following two sections focus on the hardware and software side testing for VISION. To properly meet the goals set out by the project, the team must successfully validate each system to standard tests. If standards are not met regarding these testing guidelines, changes to design must be made accordingly to properly deliver on the project's mission.

8.1 Hardware Testing

8.1.1 Guidance Testing

As guidance is at the core of VISION, its validation is critical to the validity of the system at large. VISION's design relies on audio guidance mechanisms in the form of speakers. To properly validate these, several important scopes should be examined and tested.

First, the proper output signal must be generated and be troubleshooted to an ideal signal strength that is receivable by the human ear and loud enough to be differentiated in a somewhat crowded room. To do this, the signal should be played in a room with artificial noise being introduced. If the examiner can distinctly hear the audio being generated, the waveform is validated.

The efficacy of the guidance mechanism must be placed under rigorous testing following the validation of perceivable sound. To do this, a simulated impaired user (blindfolded team member) shall be used in both the case of positioning and orientation guidance. To validate positioning guidance, the user should be able to follow basic commands from the speaker array. The efficacy of these commands can be examined on both their validity in general positioning, their ability to cease use after arrival, and the accuracy of the positioning within the proposed margin of error of six inches. Examining the orientation mechanism will then follow this stage, and will involve validating the expected output signal, proper speaker outputs, and that the user can be within the 15° margin of error.

The end goal of this validation scheme is that the user is within close enough proximity to an accurate shot the actions can be taken by the SCRATCH system to commence final guidance and deliver upon the promise of the design: making accurate shots. Test runs will be deployed to see how many successful hand offs are conducted. Finally, the most crucial test will be conducted in seeing how accurate the design is at the end of the day. How many successful implementations can be done from initial guidance all the way to successful shots. If this number reaches nontrivial values, the guidance system for VISION will be considered a success.

8.1.2 BLE Testing

As earlier mentioned in the research section, the biggest uncertainty with our localization goals is the variability of RSSI readers from BLE and the accuracy that can be obtained from using trilateration as the main means of detecting exact positions within a small margin of error. Hence, a lot of testing will come into play both on the software and on the hardware side to determine how variable the readings obtained are. Further work will need to be conducted to determine how much more accurate the measurements can be.

To begin with, testing scenarios between the ESP32 and one of Bluecharms BLE beacons is conducted. Testing will be conducted using the BC08 iBeacon, which is the non-USB one outlined in an earlier section The ESP32 module being used is the ESP32-WROOM-32 described earlier advertised as well suited for Wi-Fi and Bluetooth/Bluetooth LE-based connectivity applications and providing a solid dual-core performance. The module is part of the ESP32 Devkit V1.

As initial testing, code will be run on the ESP32 connected to a computer and constantly reading RSSI readings from the BLE beacon at its default advertisement interval and at a set direct distance from the beacon. We will use the following formula described before in the research section to compute the distance by varying different values of *n* under reasonable values from 2 to 4: 2 corresponding to large open rooms and 4 corresponding to heavily furnished rooms.

$$d = 10^{(TX - RSSI)/10n}$$

Using this as an initial testing mechanism will assist in determining how much variability there is and a finding suitable range of values for *n* which will work regardless of the room the pool table will be ultimately placed in. The RSSI values will be obtained for a specific period of time of 5 to 10 seconds and will be averaged out to use in our formula.

After obtaining a suitable value of *n* to be used, the value of *n* must be checked to see how well this value works at different distances as well as the variation in RSSI readings at different distances.

To collect data, a user will change the distance between the ESP32 and the ibeacon while staying within the 7 feet standard pool table distance range longitudinally, vertically and diagonally. Someone will record the different RSSI values for a specified period of time of about 5-10 seconds under the default advertisement interval that the iBeacon comes in, compute the mean of the RSSI values, standard deviation, and finally use the same formula with our computed value of *n*. The next test conducted will determine the effect of changing the advertisement interval. iBeacon specifies that a lower advertisement interval while being more power draining will provide better results when averaged. Tests will be conducted to vary *n* from the default advertisement interval to the lowest possible advertisement interval and compute the average of values, calculate the distance and percentage error to determine the drawback or any advantage of lowering the value. It should be noted that this might not be as large of a factor as we expect since the RSSI values being read also depend on how often the program loops through and read values on the ESP32. The main assumption here is that on the software side, the program is reading RSSI values as fast as possible and only testing what happens on the hardware side.

The last test needed is to determine if there are any adjustments needed based on devices interfering with the testing. The main assumption here is that any other Bluetooth device might affect the RSSI readings and this theory will test it out by having different Bluetooth enabled devices around the ESP32. This will be a short test and will only really be done under heavy interference around the device mimicking a lot of pool players being actively on their phones during a pool game.

The rest of the testing will be done on the software side. Once any pitfalls on the Bluetooth beacon have been discovered, the real options are either repeating these tests with a different beacon or reader module or trying to correct or smooth the readings from the RSSI values before using them for the computation. The trilateration algorithm and how the three beacons interact together will also need to be tested. The code written to transfer the values wirelessly from the ESP32 to the computer instead of it being a direct USB connection will also need to be tested. This test will also allow for the testing of the button on the hardware side, battery powering the ESP32 and the effect of any latency in the Wi-Fi communication and simulate how long the user would have to wait in place realistically as their position is being localized between shots. Many of these considerations are software problems and will be discussed in the software section.

8.2 Software Testing

8.2.1 Shot Selection Algorithm Testing

In order to ensure that the shot selection algorithm produces consistent and valid results, several test cases will be run to ensure the user is not prompted to do a task which is either impossible or illogical. Many of the test cases will correspond with the section for edge cases. The testing will feature three approaches.

- Programmatic Testing - Testing will be done after any change to the code is made, results will come back quick and will give rapid feedback on any breaking changes.
- Simulation validation - Visually verify that the results from the shot selection algorithm make sense from the display. This should be done after any major changes to the system.
- Physical Testing - Verify that the shot selection algorithm produces shots which are comfortable and realistic to attempt. This should be attempted sparingly, but at least one successful run should be made before any overall system tests are performed.

Testing shot selection: There will be several test cases that have an obvious correct answer. Ensuring that a correct decision is made on an obvious table state is of extreme importance and points to a reliable algorithm. The testing will feature a simulation that goes along with the shot selection, the table state will be provided to both the simulation and the shot selection algorithm. A success of the test case will be when the simulation executes the shot selection algorithm and makes the desired ball. The following test cases will be verified:

Will execute six tests for each case, one for every pocket

1. Ball and cue lined up in front of a pocket.
   Pass: Shot made
   Fail: Scratch or no made shot
2. Simple bank shot
   Pass: Shot made
   Fail: Scratch or no made shot
3. No easily makeable shot
   Pass: No scratch
   Fail: Scratch

Physical Limitation Tests: These tests focus on ensuring that the physical limitations of the player are respected in order to give achievable shots. The test cases should cover the previous shot selections as well, as a test passing for shot selection but not being possible is a poor indicator of our software quality. The following test cases will be verified:

Pass: The shot conforms to physical limitations as listed above
Fail: The shot fails to conform to physical limitations

1. Shot selection tests for right handed player
2. Shot selection tests for left handed player

8.2.2 Computer Vision Software Testing

The computer vision system is the initial input to the project, so the system must function accurately so errors are not propagated to other systems. The difficulty in testing the computer vision system stems from the nature of billiards itself. There are an infinite number of ways that the billiard balls can arrange themselves on the table, so it is not feasible to test every possible input configuration. The testing procedures will include the most common scenarios that a player might encounter and a few edge cases. As the project progresses, necessary test cases will be added to ensure that the computer vision system is functioning properly. This section outlines some of the most prevalent scenarios that must be tested but are by no means comprehensive of all possible input scenarios.

Testing the Billiard Table Isolation: The billiard table isolation feature of the computer vision system is the simplest feature to test. This feature is responsible for outlining the playable region of the billiard's table from the input image. The output for tests related to this feature should all have nearly the same output. The output should include a rectangular contour outlining the playable region and two populated CSV files. The output CSV file will contain the localized coordinates of the six pockets and the localization CSV file will contain the localization values and the minimum and maximum x-coordinates and y-coordinates. Although the outputs of this system may not be the same for every iteration, the values contained in the CSV files should be relatively similar.

Testing the billiard table isolation feature will be done in two stages. The first stage will be visually inspecting the contour outlining the playable region. The contour should be a rectangle that borders the playable region and should not extend outside of the playable region. There currently does not appear to be a way to automatically test this output with a high level of accuracy. The second stage includes verifying the CSV file outputs. Although the exact pixel values may fluctuate between iterations, the values should not vary by more than ten pixels in any given direction. For this reason, once well-established values for the pockets and minimum and maximum coordinates are known, automatic testing can be implemented to ensure the values in the CSV files fall within a normal range.

The state of the billiard table will not impact the output of this feature significantly, so the number and color of balls present for these inputs are not important. Certainly, test cases will be included that have a varying number of balls present on the table, but these types of tests should not drastically impact the outcome. The more significant input images for testing the billiard table isolation feature are using various lighting conditions, having the cue stick present, and having a user standing around the table. All of these scenarios are certainly possible and the computer vision system should be able to function regardless of these obstructions. Even though the billiard's table will mostly be stationary, it is still possible that lighting conditions can change and the system should still be able to function properly. Although an input image should never be taken when a user is leaning over the table, is it possible that the user is standing beside the table. The presence of a user or a pool cue should not impact the system's ability to identify the playable region of the table.

Testing for Finding the Billiard Balls: The feature responsible for finding all of the billiard balls on the table will be the most complicated feature to test. This feature includes detecting all the billiard balls in the image, determining the coordinates of the billiard balls, and determining the color of the billiard balls. This position of the computer vision system is also responsible for identifying and ignoring false positives in the input image. The output of tests related to this feature will be the information appended to the output CSV file. When this portion of the computer algorithm is run, the locations of the six pockets will already be included in the output file. This section will append the type of ball found (game ball, cue ball, or black ball), the localized x-coordinate, and the localized y-coordinate for every billiard ball in the input image.

Before discussing how to create unit tests for this feature, a brief discussion on testing for the minimum and maximum radius is needed. Section 3.2.2 describes utilizing the parameters available in OpenCV's Hough Circle Transform to specify the minimum and maximum radius. To determine the minimum and maximum radius, other built-in

OpenCV features can be used. By running the Hough Transform without any radius requirements, all of the circles in the image will be discovered. The discovered contours can be manually iterated and highlighted so each contour can be verified for correctness. The area of all of the correct contours can then be found by using an OpenCV area method. Once a substantial amount of samples have been collected, the average radius, in pixels, can be extracted from the area measurements. An appropriate radius threshold can then be set.

Testing this feature is ensuring that the output CSV file is updated properly to reflect the current state of the billiard table. To ensure that the feature is working properly, simple testing will be conducted and more complex scenarios will be added. Simple tests of the system include capturing input images where billiard balls are on the table in a variety of configurations. The output file should accurately represent the number, color, and location of the types of balls on the table. It will be important to consider lots of different combinations of inputs. Once the basic scenarios are ensured to be working properly, more complex scenarios can be added. Important scenarios to consider would be when the white ball is not present, the black ball is not present, neither the black ball nor the white ball is present, and when no balls are present. Other more complex scenarios are when two or more balls are touching, the cue stick is present in the input image, and when there are circular objects in the input image that are too small or too big to be billiard balls. All of these scenarios should also be considered in different lighting conditions to ensure that the accuracy of the computer vision system is not diminished by different lighting conditions.

A set of automated unit tests will be created by capturing many input images representative of the previously described testing scenarios. Generating a suite of unit tests will ensure that the system is functioning as expected. These unit tests will have a verified output CSV file associated with each input image so that any changes to the computer vision system can quickly be verified against an established set of tests. Creating such a testing environment is important because it will allow for changes to the project to be verified quickly, without having to manually test the new modifications.

8.2.3 Feedback System Software Testing

Testing the Shot Result Feedback: Testing the shot result logic of the feedback system is one of the most important features to test in the project. The shot result subsystem should be able to take the previous and current state of the billiard table and determine the outcome of a player's shot. This subsystem is straightforward and can be easily tested. The inputs for the feedback are two CSV files originating from the computer vision system. One of the CSV files is the previous state of the billiard table and the other CSV file is the current state of the billiard table. It is possible to create test CSV files representative of all possible scenarios the computer vision system can output. Once created, these input files will form a test suite used against the expected output to ensure that the system is functioning properly.

The actual testing of the shot result feedback consists of checking if the cue ball is present, if the eight ball is present, how many green balls are present, and how many blue balls are present. If the eight ball is present, then the user has either won or lost the game. The deciding factor is if the player has any game balls left on the table. If the eight ball is not present, the user will continue playing and has either not sunk a ball, sunk their game ball, or sunk an opponent's game ball. All of these scenarios are predictable and can be tested easily with custom CSV input files.

8.2.4 Localization Software Testing

This section reiterates some of the problems with BLE and goes over how we will test out the final system we will be using to localize the user at every point and while they are in motion. The main goal of the localization system should be that it allows the position of the user to be detected and returned to a guidance system at every time while the user is moving from their initial position to the final position. The first concern is the accuracy of the system that will be testing. In the hardware section, the theory behind determining the distance between a receiver and a transmitter enabled with BLE was tested. The final design expands upon this by using three beacons each advertising to the ESP32 and the signals received by our ESP32 are in turn used to compute the distance between the ESP32 and the beacons. The distances are in turn used to triangulate the position of our ESP32 and hence our user. Preliminary testing can be done without the need of a pool table for this system. With the code written out to perform the collection of RSSI values, transfer via Wi-Fi to a computer, and then computation of (x,y) position based on these values, the table setup can be simulated. Testing will be done by moving the ESP32 around the table in a predetermined manner and tracking if the distances computed match with the motion traced around the simulated table.

The next test will be determining how accurate the readings are by placing the ESP32 at specific points for a prolonged period of time and determining the error between the expected value and the computed value by our testing algorithm.

Other parameters to record during testing are the latency that may occur between communications and within the code itself. As is, running both BLE and Wi-Fi on the ESP32 devkit module is a heavy task for the processor. Earlier it was mentioned that the BLE beacons are able to broadcast as fast as sending one advertisement packet every 100 milliseconds or 10 every second. However, limitations on the side of the ESP32 prevent the beacons from fully taking advantage of this feature. As is, using available modules online for BLE on the ESP32, the ESP32 is only able to set a scanning time within seconds. In other words, the most scanning the ESP32 can do is scan every second. For each scan, the ESP32 is also able to modify the scanning interval and the scanning window which determines how long it will be actively looking for an advertisement packet within this time window. Specific examples available for the ESP32 set this scanning interval to 100 milliseconds and the scanning window to approximately the same value. This, in conjunction with the advertisement interval set on our beacon, should be enough to allow for detecting any of the advertisement packets sent within that second to the ESP32.

Additional latency may be incurred by adding Wi-Fi to our functionality. Tests will need to determine the latency associated with the different formats in which data is sent from ESP32 to the main computer. The main protocol under consideration is UDP described under the communication protocols section. Testing the fastest format for sending data in a way that will be either easier to transmit for the ESP32 or easier to receive for our main computer or controller will be very important.

The last latency issue that needs to be tested is the button press implementation that will determine when the user is ready to move from one point to another. One of the things to test for this issue is determining if it is a necessary addition. Its main usage will be giving VISION a set start point to run the code for localization. An alternative would be to simply run the code constantly and allow the guidance system to use the output of the localization system at any time for user guidance. Both options would be applicable mainly dependent on the guidance algorithm or what seems more user friendly at the end. If implemented either way, the button might be used for other features within this project such as allowing a reset of the localization/guidance system for example.

Other components of the localization system pertaining to hardware that were not mentioned in that section that would not require any additional testing but are worth mentioning are the batteries we use for powering the ESP32, the battery life of the beacons (considering they can be turned on and off to save power), the 3D model and print of the case within which the ESP32 will lie in and general wiring.

8.3 User Testing

To evaluate the success of VISION and SCATCH, a visually impaired user should be navigated around the billiards table and able to successfully complete a clear shot. The success of the projects largely depends on a user's ability to complete a shot. If the system created by VISION and SCRATCH can allow a user to sink a billiard ball, the system will be considered successful.

From VISION's perspective, the first benchmark is being able to properly capture the state of the billiard's table and represent the table state computationally. The table representation should also be able to produce a reasonable shot selection with the help of the billiards artificial intelligence system. This process is not easily verifiable and will require the VISION team to manually verify the shot. The table representation will need to be verified to ensure that the representation accurately reflects the state of the table. The shot selection will need to be verified to ensure that the artificial intelligence algorithm selects a shot that is feasible and guides the user to progress towards winning the game. These verifications will be performed by testing the system with an actual user and verifying VISION's decisions in real-time.

The second benchmark of VISION is being able to locate and guide the user around the billiards table. The user's location should be checked against the location of the user that

VISION reports to the system. If the user is within the allowable distance of the user localization system, the system will be deemed a success. The user guidance should be able to guide a user around the billiards table from a starting location to a final location. The system will be tested by guiding a user from some starting location to some predetermined final location. If the user is able to be guided to the final location within the specified margin of error, the user guidance system will be considered successful.

Overall, there is no automatic way to test the effectiveness of VISION. Individual test cases will be designed for each subsystem to validate the subsystems basic behavior. Success during individual testing does not correlate to success of the overall project. The project can only be validated by testing the entire system and verifying the system's results in real-time. Subsystem testing will help to eliminate major subsystem issues, but the true test of VISION will occur when all of the subsystems are integrated.

9. ADMINISTRATIVE CONTENT

9.1 Project Budget

VISION is a large project that requires a significant amount of hardware and software components. As shown in the table below, the project requires a billiards table, Jetson Nano, camera, multiple BLE beacons, and other costly hardware. To account for the large amount of technology needed, the team has set a budget of $800 ($200 per team member). The budget is an upper bound of what the team believes is needed for someone to recreate this project.

9.1.1 Bill of Materials

Table 9.1 lists the materials, quantity, and associated cost for the materials needed to implement VISION.

Component	Quantity	Unit Cost	Total
Pool Table	1/2	$450	$225
Anker Powerconf c200	1	$50	$50
ESP Microcontrollers	2	$15	$30
Bluetooth Beacons	3	$20	$60
PCB Testing Parts	1	$20	$20
PCB Final Assembly Parts	1	$40	$40
Jetson Nano 4GB Development Kit	1	$200	$200
Speakers	12	$2	$24
Monitor	1	$40	$40
Total			$689

Table 9.1: Bill of Materials

9.1.2 Project Financing

The table above is a comprehensive list of the most critical components for VISION. The pool table will be shared with the SCRATCH (group #17) project, meaning the team is only responsible for half of the cost of the pool table. Although the price of the project is within the $800 project budget, there are opportunities to reduce the overall cost. Due to supply chain shortages, most high-power processors (Jetson Nano, Google Coral Dev Board, Raspberry Pi) are not in stock and are subject to third-party resale prices. The team is currently reaching out to suppliers to try and obtain a board at retail price. VISION and SCRATCH are actively seeking sponsorship and outside funding for the project. In the worst case, the members of VISION will split the costs of the project between themselves.

9.2 Milestones

VISION is a complex project requiring many different systems to integrate together for a user to play a game of billiards. For this reason, the members of VISION used the summer prior to taking Senior Design 1 to complete the project brainstorming. The goal was for the team to start the documentation process as soon as classes resumed so there would be sufficient time to research the design. There are no complete projects for VISION to be based upon, so the group wanted to ensure adequate time to resolve any issues arising while conducting research.

The timelines discussed below account for any research compilations that may be discovered. The milestones of VISION will ideally be completed before the anticipated end dates so that the documentation can be submitted before the due date. Although the focus of Senior Design 1 is the research and documentation of the project, the team plans to begin preliminary testing to show that the ideas being researched are feasible. Proof of concept testing will be conducted by each member in their respective area of focus alongside their project research. After the project report has been submitted it will be used for further testing. Ideally, system integration can be performed as soon as the team moves into Senior Design 2. For a more detailed schedule of VISION's goals, view tables 9.2, 9.3 and 9.4.

Task	Start Date	Anticipated End Date	Duratio n
Project Brainstorming	Summer	Summer	0 weeks
Project Scope Finalized (Finalize big picture design and what the end goal is)	08/22/2022	08/26/2022	1 week
Individual Research Begins (Begin breaking the project into smaller subsections such as CV or AI)	08/22/2022	09/02/2022	2 weeks
Initial Design Document (Based upon the D&C documents)	08/22/2022	09/05/2022	1.5 weeks
30-Page Milestone (General system design, project motivation, project goals, project concepts)	08/22/2022	09/09/2022	3 weeks
60-Page Milestone (Independent technology research, system requirements, part ideas/availability)	09/10/2022	09/30/2022	3 weeks
90-Page Milestone (Independent technology research, system communication)	10/01/2022	10/21/2022	3 weeks
120-Page Milestone (System testing, PCB design, PCB testing, citations)	10/22/2022	11/11/2022	3 weeks
Group Review: Final Draft	11/14/2022	11/18/2022	1 week

Table 9.2: Senior Design 1 Project Documentation Milestones

Task	Start Date	Anticipated End Date	Duration
Individual System Design (Create some proof of concept design in hardware or software)	09/05/2022	10/02/2022	4 weeks
Individual System Testing (Develop and demonstrate the proof of concept design to the team)	10/03/2022	10/30/2022	4 weeks
Breadboard Prototyping (Finalize what the PCB will do and breadboard the design)	10/31/2022	11/21/2022	3 weeks (Assuming we can get parts in time)
PCB Design / Ordering (Design the PCB in Eagle and order from a reputable PCB company)	11/22/2022	12/12/2022	3 weeks (Assuming we can get parts in time)

Table 9.3: Senior Design 1 Project Design Milestones

Task	Start Date	Anticipated End Date	Duration
PCB Testing (Test all of the PCBs to ensure they work properly)	01/09/2023	01/23/2023	2 weeks
System Integration / Testing (Begin integrating the individual systems together in the main code)	01/24/2023	02/13/2023	4 weeks
Practice Project Demo (Go through a mock project demonstration to ensure everything is functioning)	02/14/2023	02/27/2023	2 weeks
Finalize Documentation (Final edits and construction of the documentation)	02/28/2023	03/13/2023	2 weeks
Practice Final Presentation	03/14/2023	03/20/2023	1 week
Final Presentation	TBD	TBD	TBD

Table 9.4 Senior Design 2 Project Design Milestones

The tables above is a tentative schedule with emphasis placed on the documentation milestones over the design milestones. The schedule should not need modification because of the team's commitment to brainstorming the project over the summer. If any significant problems develop while researching VISION, two weeks are currently unaccounted for that can be allocated to any milestone deadline as needed. The work done during the design milestones will guide the selection of hardware and software in the document as members can discover what will and will not work for the project

10. PROJECT SUMMARY & CONCLUSION

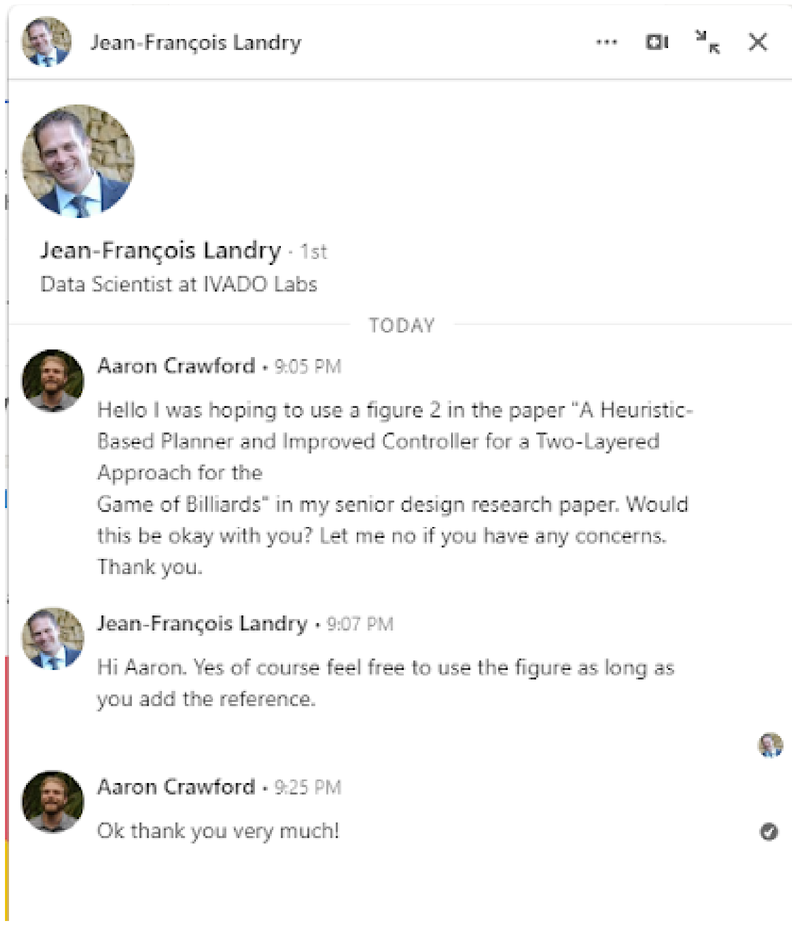
VISION is progressing well throughout the Senior Design 1 semester. The VISION team has reviewed many different types of applicable technology and developed a better understanding of what technologies will be applicable during project design in Senior Design 2. Furthermore, the team has developed a hardware and software design plan that has shown positive results in preliminary testing.

One of the largest issues that VISION, and other projects, must overcome is the remaining problems in the supply chain. Many parts that VISION would like to use are either unavailable or significantly more costly due to having to pay third-party prices. In addition to product unavailability, shipping times, especially from international sources, is still slower than pre-pandemic times. VISION is overcoming these difficulties by acquiring parts now so that there is no delay to design in the spring semester.

VISION has been able to acquire a billiards table, a web camera for computer vision, a reliable billiards artificial intelligence program, Bluetooth beacons, ESPs, speakers, and a Jetson Nano. Although the VISION team will still need to wait sometime before being able to acquire the PCB, many of the core components of the project have already been acquired.

The VISION team's dedication to technology exploration in Senior Design 1 has allowed the team to discover, discuss, and solve many design issues related to the project's implementation. With a wealth of new knowledge on the subject and many of the necessary components acquired, the VISION team is looking forward to implementing the project in Senior Design 2.

Appendix A: Copyright Permissions



Request for Shot Planner Diagram (Figure 3.1)

Permissions for "An Evolving Signature Recognition System" Image

Report

Noah Harney Aug 11, 2022

Hello Dr. Jayasiri,

My name is Noah Harney and I am an undergraduate student at the University of Central Florida (Orlando, Florida) and am working on my senior design project. I am researching and developing a computer vision algorithm and must write a design document for the project. I came across your paper "An Evolving Signature Recognition System" and found it very useful. I would like to request permission to use an image from the paper in my design document. I would like to include Fig 1. Histogram based thresholding in my document. The document is for educational purposes at my university. I have attached a screenshot of the image below for your convenience.

Are you okay with me using your image (attached below) in my design documentation? I will cite your work in my design document.

Best,
Noah H.

An_Evolving_Signature_Recognition_System.png

Awantha Jayasiri to you Aug 15, 2022

Hi Noah,
No worries, Go ahead and use it. Happy to help.

Request and Permission for Image of Thresholding Distribution (Figure 3.4)

Name *(Required)*

Noah Harney

Email *(Required)*

nharney1@knights.ucf.edu

Message *(Required)*

Hello Adrian,

I found your article "OpenCV Thresholding (cv2.threshold)" useful for a senior project I am working on. I am using OpenCV to create a computer vision system for helping individuals dealing with visual impairments play billiards. I was wondering if I would be allowed your figure 6 in my paper (giving credit to you and your article of course) to help illustrate some of the concepts I am talking about.

The exact article I am referring to is:
https://pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/

Best,
Noah H.

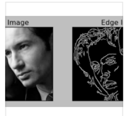
SEND

Request for Image of Thresholding (Figure 3.5)

Permissions for "Canny Edge Detection" Image

NH Noah Harney <nharney1@Knights.ucf.edu>
8/11/2022 2:17 PM

To: contactus@bogotobogo.com



Hello Dr. Hong,

My name is Noah Harney and I am an undergraduate student at the University of Central Florida (Orlando, Florida) and am working on my senior design project. I am researching and developing a computer vision algorithm and must write a design document for the project. I came across your article "Canny Edge Detection" and found it very useful. I would like to request permission to use an image from the article in my design document. The document is for educational purposes at my university. I have attached a screenshot of the image below for your convenience.

Are you okay with me using your image (attached below) in my design documentation? I will cite your work in my design document.

I have included a link to the article I am referencing for your convenience:
https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Canny_Edge_Detection.php

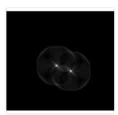
Best,
Noah H.

Request for Canny Edge Detection Image (Figure 3.6)

Permissions for "Circle Hough Transform" Image

NH Noah Harney <nharney1@Knights.ucf.edu>
8/11/2022 2:24 PM

To: utkarsh@utkarshsinha.com



Hello Dr. Sinha,

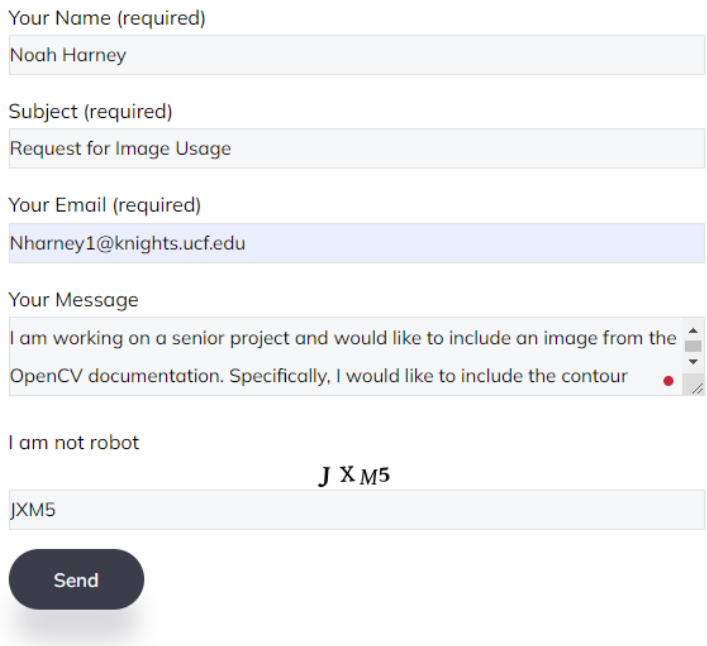
My name is Noah Harney and I am an undergraduate student at the University of Central Florida (Orlando, Florida) and am working on my senior design project. I am researching and developing a computer vision algorithm and must write a design document for the project. I came across your article "Circle Hough Transform" and found it very useful. I would like to request permission to use an image from the article in my design document. The document is for educational purposes at my university. I have attached a screenshot of the image below for your convenience.

Are you okay with me using your image (attached below) in my design documentation? I will cite your work in my design document.

I have included a link to the article I am referencing for your convenience: https://aishack.in/tutorials/circle-hough-transform/

Best,
Noah H.

Request for Hough Circle Transform Image (Figure 3.7)



Your Name (required)

Noah Harney

Subject (required)

Request for Image Usage

Your Email (required)

Nharney1@knights.ucf.edu

Your Message

I am working on a senior project and would like to include an image from the OpenCV documentation. Specifically, I would like to include the contour

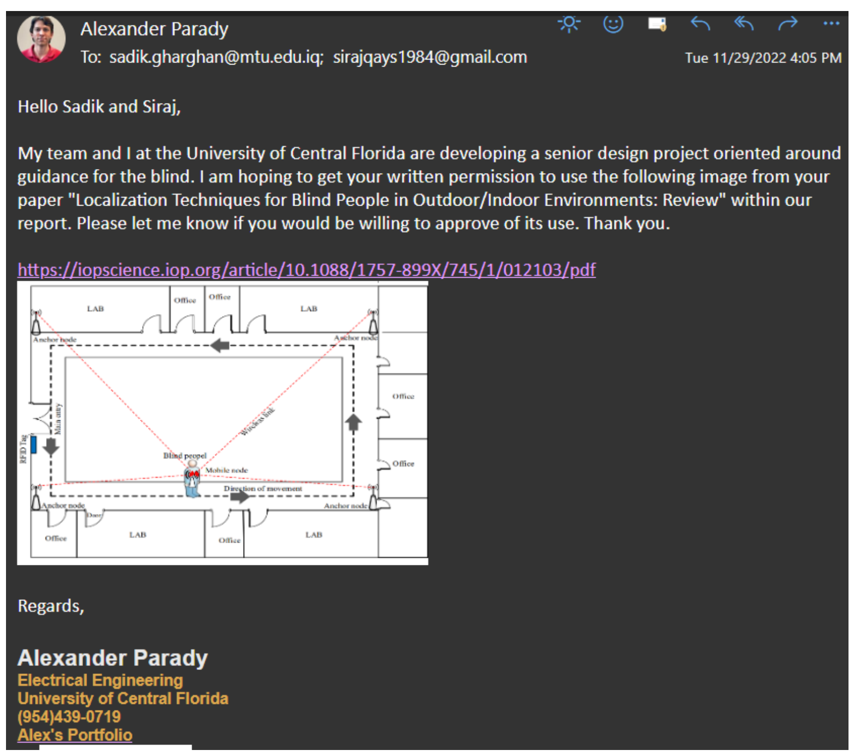
I am not robot

J X M5

JXM5

Send

Request for Douglas-Peucker Algorithm (Figure 3.8)



Alexander Parady

To: sadik.gharghan@mtu.edu.iq; sirajqays1984@gmail.com

Tue 11/29/2022 4:05 PM

Hello Sadik and Siraj,

My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following image from your paper "Localization Techniques for Blind People in Outdoor/Indoor Environments: Review" within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://iopscience.iop.org/article/10.1088/1757-899X/745/1/012103/pdf

Regards,

Alexander Parady
Electrical Engineering
University of Central Florida
(954)439-0719
Alex's Portfolio

Request for Previous System Indoor Localization Design (Figure 3.9)

Arsene LT <landryarsene619@gmail.com> 11:39 PM (7 minutes ago)

to rbis.americas

Hello,

I am a student at the University of Central Florida working on an end-of-curriculum senior design project where part of it involves writing a document with a section on research for the said project, and might involve getting sources and pictures online. One of which is the following picture taken from the webpage for the said product. Would it possible to use this for our document?

Best regards,

--

Arsene Landry Tatke,

352-328-8686

UCF | Electrical Engineering Student

Request for Image of Avery Dennison's AD-172u7 Inlays (Figure 3.10)

Ashlie Whitten <ashlie.whitten@averydennison.com> Wed, Nov 30, 9:14 AM (2 days ago)

to me

Hi Arsene,

You may use this image for your school project, thanks for asking on rights and permission and best of luck!

Ashlie

Ashlie Whitten

Director, Global Marketing and Communications

Avery Dennison - Identification Solutions

AVERY DENNISON

YouTube LinkedIn

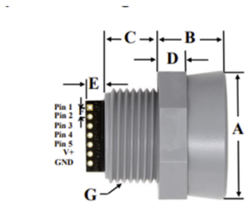
averydennison.com

Approval for Image of Avery Dennison's AD-172u7 Inlays (Figure 3.10)

Arsene LT <landryarsene619@gmail.com> 11:07 PM (39 minutes ago)
to info@maxbotix.com

Hello,

I am a student at the University of Central Florida working on an end-of-curriculum senior design project where part of it involves writing a document with a section on research for the said project, and might involve getting sources and pictures online. One of which is the following picture taken from the datasheet for the said product. Would it possible to use this for our document?



A	1.37" dia.	34.7 mm dia.
B	0.70"	17.9 mm
C	0.57"	14.4 mm
D	0.31"	7.9 mm
E	0.23"	5.8 mm
F	0.1"	2.54 mm
G	3/4"-14 NPS	
H	1.032" dia.	26.2 mm dia.
I	1.37"	34.8 mm
Weight, 1.23 oz., 32 grams		

Values Are Nominal

Thank you for your time.

With best regards,

Request for image of Model and Dimensions of Compact Housing HRXL-MaxSonar Model (Figure 3.13)

Joe Pickett <jpickett@maxbotix.com> Wed, Nov 30, 8:49 AM (2 days ago)
to me, info@maxbotix.com

Yes, you may use this image for your project.

Please footnote your project appropriately advising that the source of the photo was MaxBotix Inc.

Thanks.

Joe

We would love your feedback on how we're doing.
Leave us a review

Joseph Pickett

Chief Financial Officer

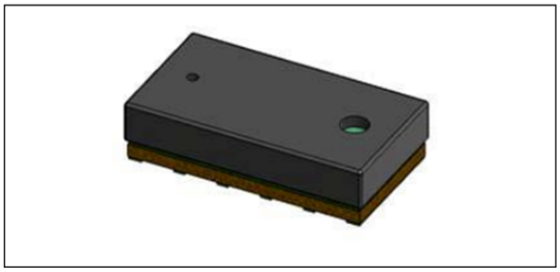
MaxBotix Inc.
Phone: +1 (218) 454-0766
Direct: +1 (218)-454-7333
Fax: +1 (888) 380-3977
Email: jpickett@maxbotix.com
Web: www.maxbotix.com

Approval for image of Model and Dimensions of Compact Housing HRXL-MaxSonar Model (Figure 3.13)

Arsene LT <landryarsene619@gmail.com> 11:36 PM (10 minutes ago)
to infocontact

Hello,

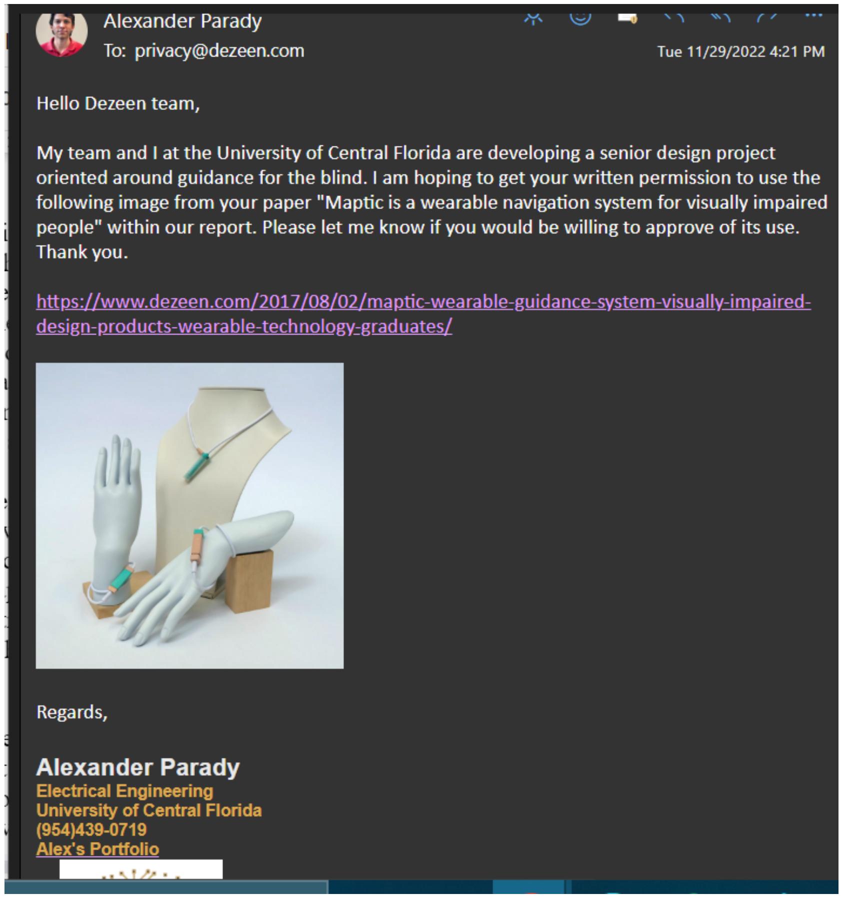
I am a student at the University of Central Florida working on an end-of-curriculum senior design project where part of it involves writing a document with a section on research for the said project, and might involve getting sources and pictures online. One of which is the following picture taken from the datasheet for the said product. Would it possible to use this for our document?



Thank you for your time.

With best regards,
--
Arsene Landry Tatke,
352-328-8686

Request for image of VL53L0X Time-of-Flight Ranging and Gesture Detection Sensor (Figure 3.14)



Alexander Parady
To: privacy@dezeen.com
Tue 11/29/2022 4:21 PM

Hello Dezeen team,

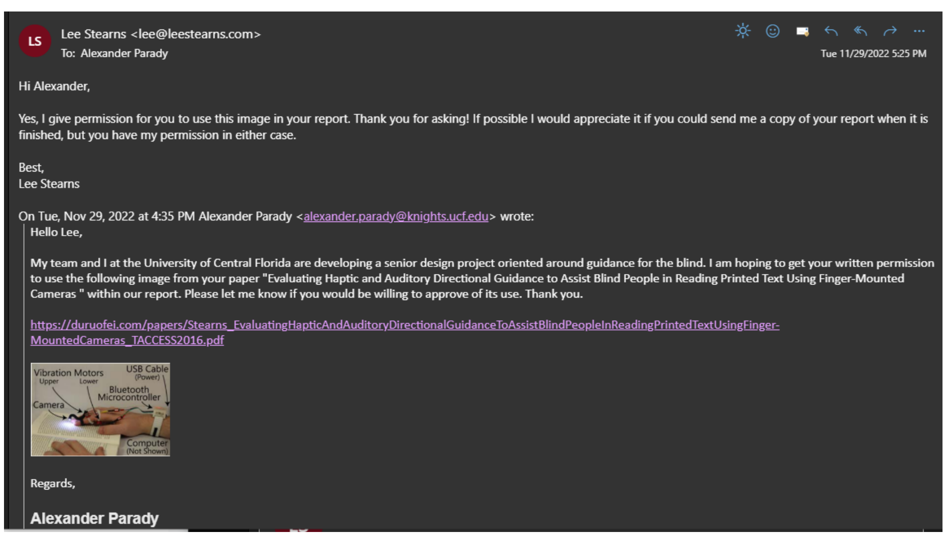
My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following image from your paper "Maptic is a wearable navigation system for visually impaired people" within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://www.dezeen.com/2017/08/02/maptic-wearable-guidance-system-visually-impaired-design-products-wearable-technology-graduates/

Regards,

Alexander Parady
Electrical Engineering
University of Central Florida
(954)439-0719
Alex's Portfolio

Request for Maptic Haptic Feedback Apparatus (Figure 3.17)



Lee Stearns <lee@leestearns.com>
To: Alexander Parady
Tue 11/29/2022 5:25 PM

Hi Alexander,

Yes, I give permission for you to use this image in your report. Thank you for asking! If possible I would appreciate it if you could send me a copy of your report when it is finished, but you have my permission in either case.

Best,
Lee Stearns

On Tue, Nov 29, 2022 at 4:35 PM Alexander Parady <alexander.parady@knights.ucf.edu> wrote:

Hello Lee,

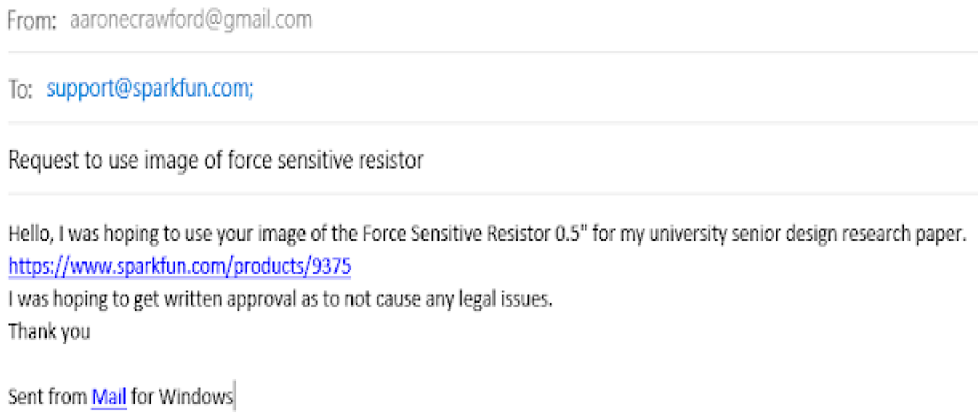
My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following image from your paper "Evaluating Haptic and Auditory Directional Guidance to Assist Blind People in Reading Printed Text Using Finger-Mounted Cameras " within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://duruofei.com/papers/Stearns_EvaluatingHapticAndAuditoryDirectionalGuidanceToAssistBlindPeopleInReadingPrintedTextUsingFinger-MountedCameras_TACCESS2016.pdf

Regards,

Alexander Parady

Request for HandSight Haptic Feedback Apparatus (Figure 3.18)



From: aaronecrawford@gmail.com

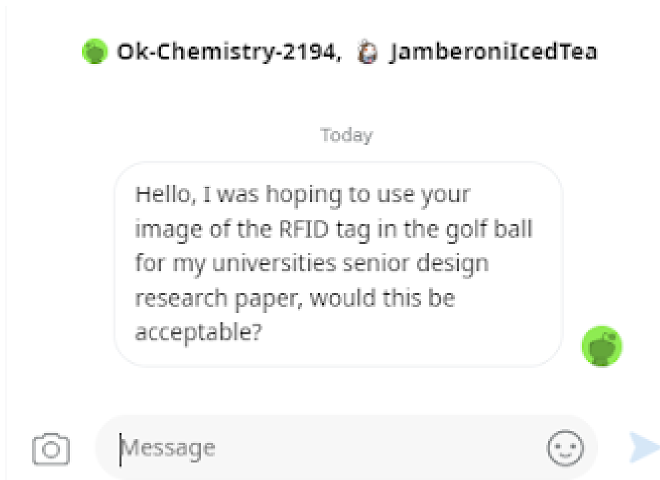
To: support@sparkfun.com;

Request to use image of force sensitive resistor

Hello, I was hoping to use your image of the Force Sensitive Resistor 0.5" for my university senior design research paper.
https://www.sparkfun.com/products/9375
I was hoping to get written approval as to not cause any legal issues.
Thank you

Sent from Mail for Windows

Approval for Image of Force Sensitive Resistor from Sparkfun (Figure 3.20)



Ok-Chemistry-2194, JamberoniIcedTea

Today

Hello, I was hoping to use your image of the RFID tag in the golf ball for my universities senior design research paper, would this be acceptable?

Message

Request to Use Image of RFID Tag in Golf Ball from Reddit User (Fig 3.21)

Alexander Parady
To: sales@maxiaids.com

Tue 11/29/2022 4:40 PM

Hello Maxi-Aids team,

My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following image from your product page "Tek Pal Tactile Low Vision TV Remote Control " within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://www.maxiaids.com/tek-pal-tactile-low-vision-tv-remote-control

Regards,

Alexander Parady
Electrical Engineering
University of Central Florida
(954)439-0719

Request for TV Remote for the Visually Impaired (Figure 3.23)

To: Customer Service <Custserv@bluewaveproducts.com> Thu 12/1/2022 9:20 PM

Hello Blue Wave Team

My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following product image within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://bluewaveproducts.com/products/fairmont-6-ft-portable-pool-table-black-with-blue-felt

To: Support <Support@RACKPoolTables.com> Thu 12/1/2022 9:25 PM

Hello Rack Team

My team and I at the University of Central Florida are developing a senior design project oriented around guidance for the blind. I am hoping to get your written permission to use the following product image within our report. Please let me know if you would be willing to approve of its use. Thank you.

https://www.rackpooltables.com/product/rack-crux-55-in-folding-billiard-pool-table-blue/

Request for Blue Wave's Fairmount Table (Top) & Rack's Crux 55 Table (Bottom)
(Figure 5.1)

Appendix B: Code Segments

Dash

```
import dash
import dash_html_components as html
import dash_core_components as dcc

app = dash.Dash(__name__)

app.layout = html.Div([dcc.Graph(id="main-graph",
figure=fig)])

if __name__ == '__main__':
   app.run_server(debug=True)
```

Streamlit

```
import streamlit as st
st.write(fig)
```

Dash vs Streamlit: Setting up a locally running app

Appendix C: References

Accessible Pedestrian Signals. http://www.apsguide.org/chapter7_adjustments.cfm.

Adaptive-Vision. "Template Matching." *Adaptive-Vision*, https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html. Accessed 11 November 2022.

"Assistive Technology for the Blind (AT)." *Mass.gov*, https://www.mass.gov/service-details/assistive-technology-for-the-blind-at. Accessed 29 November 2022.

"Audio guidance system for blind." https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8203710&tag=1.

Authentise. "Detecting Circular Shapes Using Contours." *Authentise*, 18 April 2016, https://www.authentise.com/post/detecting-circular-shapes-using-contours. Accessed 11 November 2022.

BogoToBogo. "OpenCV 3 Canny Edge Detection - 2020." *BogoToBogo*, https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Canny_Edge_Detection.php. Accessed 11 November 2022.

Breckon, Toby, and Chris Solomon. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley, 2011.

Cannizzaro, Davide. "A Comparison Analysis of BLE-Based Algorithms for Localization in Industrial Environments." ,, 26 February 2022, https://www.researchgate.net/publication/338241733_A_Comparison_Analysis_of_BLE-Based_Algorithms_for_Localization_in_Industrial_Environments. Accessed 30 November 2022.

Data Carpentry. "Thresholding – Image Processing with Python." *Data Carpentry*, https://datacarpentry.org/image-processing/07-thresholding/. Accessed 11 November 2022.

"Datasheet for the HRXL-MaxSonar-WR sensor line." *MaxBotix Inc.*, https://www.maxbotix.com/documents/HRXL-MaxSonar-WR_Datasheet.pdf. Accessed 30 November 2022.

Davide Cannizzaro Politecnico di Torino. “A Comparison Analysis of BLE-Based Algorithms for Localization in Industrial Environments.” 26 February 2022, https://www.researchgate.net/publication/345670871_Indoor_Navigation_System_using_BLE_and_ESP32. Accessed 30 November 2022.

Digilent Corporation. “UART.” *Digilent Reference*, 29 October 2012, https://digilent.com/reference/learn/fundamentals/communication-protocols/uart/start. Accessed 11 November 2022.

“Ensure that the remote control can be used without requiring sight.” https://universaldesign.ie/technology-ict/archive-irish-national-it-accessibility-guidelines/digital-tv-equipment-and-services/guidelines-for-digital-tv-equipment-and-services/remote-controls/ensure-that-the-remote-control-can-be-used-without-requiring-sig. Accessed 1 December 2022.

“Evaluating Haptic and Auditory Directional Guidance to Assist Blind People in Reading Printed Text Using Finger-Mounted Cameras.” *Ruofei Du*, https://duruofei.com/papers/Stearns_EvaluatingHapticAndAuditoryDirectionalGuidanceToAssistBlindPeopleInReadingPrintedTextUsingFinger-MountedCameras_TACCESS2016.pdf. Accessed 1 December 2022.

Franklin, Dustin. “Jetson Nano Brings AI Computing to Everyone | NVIDIA Technical Blog.” *NVIDIA Developer*, 18 March 2019, https://developer.nvidia.com/blog/jetson-nano-ai-computing/. Accessed 11 November 2022.

Fusco, Giovanni, and James M. Coughlan. “Indoor Localization for Visually Impaired Travelers Using Computer Vision on a Smartphone.” *NCBI*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7643919/. Accessed 1 December 2022.

“Guidance System for Visually Impaired People.” https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9395973.

Herrman, John. “How To Extend Your HDMI Cables | HDMI Repeater.” *Popular Mechanics*, 1 April 2021, https://www.popularmechanics.com/home/how-to/a6751/how-to-extend-your-hdmi-cables/. Accessed 11 November 2022.

"Home." *YouTube*, https://iopscience.iop.org/article/10.1088/1757-899X/745/1/012103/pdf. Accessed 1 December 2022.

"Home." *YouTube*, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9395973. Accessed 1 December 2022.

ImageJ. "Hough Circle Transform." *ImageJ Wiki*, 21 September 2018, https://imagej.net/plugins/hough-circle-transform. Accessed 11 November 2022.

"Insight Into ESP32 Sleep Modes & Their Power Consumption." *Last Minute Engineers* -, https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/. Accessed 2 December 2022.

Jayasekara, Buddhika, et al. "An Evolving Signature Recognition System." *IEEE Xplore*, 2006, pp. 529-534. https://ieeexplore.ieee.org/document/4216646?arnumber=4216646.

"Jetson Nano + Intel Wifi and Bluetooth." *JetsonHacks*, 8 April 2019, https://jetsonhacks.com/2019/04/08/jetson-nano-intel-wifi-and-bluetooth/. Accessed 11 November 2022.

Kang, and Atul. "Suzuki Contour Algorithm OpenCV." *TheAILearner*, 19 November 2019, https://theailearner.com/tag/suzuki-contour-algorithm-opencv/. Accessed 11 November 2022.

Kangalow. "Jetson Nano + Intel Wifi and Bluetooth." *JetsonHacks*, 8 April 2019, https://jetsonhacks.com/2019/04/08/jetson-nano-intel-wifi-and-bluetooth/. Accessed 11 November 2022.

Keras. "About Keras." *Keras*, https://keras.io/about/. Accessed 11 November 2022.

Landry, Jean-François, et al. "A Heuristic-Based Planner and Improved Controller for a Two-Layered Approach for the Game of Billiards." *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES*, vol. 5, no. 4, 2013, pp. 325-346. *ieeexplore*, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6651845.

Lee, Socret. "Simplify Polylines with the Douglas Peucker Algorithm | by Socret Lee." *Towards Data Science*, 8 May 2021, https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1. Accessed 11 November 2022.

Liao, Peiyu, et al. “Deep Cue Learning: A Reinforcement Learning Agent for Playing Pool.” *stanford.edu*, Stanford, https://cs229.stanford.edu/proj2018/report/249.pdf.

List, Jenny. “All You Need To Know About I2S.” *Hackaday*, 18 April 2019, https://hackaday.com/2019/04/18/all-you-need-to-know-about-i2s/. Accessed 11 November 2022.

“Localization Techniques for Blind People in Outdoor/Indoor Environments: Review.” https://iopscience.iop.org/article/10.1088/1757-899X/745/1/012103/pdf.

“Maptic is a wearable navigation system for visually impaired people.” https://www.dezeen.com/2017/08/02/maptic-wearable-guidance-system-visually-impaired-design-products-wearable-technology-graduates/.

Meel, Vidushi. “YOLOv3: Real-Time Object Detection Algorithm (Guide) - viso.ai.” *Viso Suite*, Vvso.ai, https://viso.ai/deep-learning/yolov3-overview/. Accessed 11 November 2022.

Muthukrishnan. “Otsu's method for image thresholding explained and implemented – Muthukrishnan.” *Muthukrishnan*, 13 March 2020, https://muthu.co/otsus-method-for-image-thresholding-explained-and-implemented/. Accessed 11 November 2022.

Nancy Seckel. “Physics of 3D Ultrasonic Sensors.” ,, 26 February 2022, https://www.researchgate.net/publication/334784649_Physics_of_3D_Ultrasonic_Sensors. Accessed 30 November 2022.

NVIDIA Corporation. “Taking Your First Picture with CSI or USB Camera.” *NVIDIA Developer*, https://developer.nvidia.com/embedded/learn/tutorials/first-picture-csi-usb-camera. Accessed 11 November 2022.

NVIDIA Corporation. “Vision Programming Interface (VPI).” *NVIDIA Developer*, https://developer.nvidia.com/embedded/vpi. Accessed 11 November 2022.

OpenCV. “About OpenCV.” *OpenCV*, https://opencv.org/about/. Accessed 11 November 2022.

OpenCV. “Canny Edge Detection.” *OpenCV*, https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html. Accessed 11 November 2022.

OpenCV. “Color Space Conversions.” *OpenCV*, https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html. Accessed 11 November 2022.

OpenCV. “Contour Features.” *OpenCV*, https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html. Accessed 11 November 2022.

OpenCV. “Image Gradients.” *OpenCV*, https://docs.opencv.org/4.x/d5/d0f/tutorial_py_gradients.html. Accessed 11 November 2022.

OpenCV. “Object Detection.” *OpenCV*, 9 October 2019, https://docs.opencv.org/4.1.2/df/dfb/group__imgproc__object.html#ga586ebfb0a7fb604b35a23d85391329be. Accessed 11 November 2022.

Peterson, Zachariah. “Top 5 PCB Design Rules You Need to Know.” *Altium's Resource*, 21 February 2017, https://resources.altium.com/p/pcb-layout-guidelines. Accessed 1 December 2022.

Pinke, Ryan. “(Updated) Buying Guide: Comparing Field of View When Buying a Conference Room Video Camera.” *Video Conference Gear*, 22 January 2021, https://www.videoconferencegear.com/blog/updated-buying-guide-comparing-field-of-view-when-buying-a-conference-room-video-camera/. Accessed 11 November 2022.

Ragoo, Kiran, et al. “Design and development of a pool and billiards assistive device for the physically challenged.” *Disability and Rehabilitation: Assistive Technology*, vol. 14, no. 6, 2019. *tandfonline*, https://doi.org/10.1080/17483107.2018.1467974.

Ramirez, Ramiro, and Chien-Yi Huang. “A Practice of BLE RSSI Measurement for Indoor Positioning.” *NCBI*, 30 July 2021, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8347277/. Accessed 30 November 2022.

Raspberry Pi. “Raspberry Pi Documentation - Camera.” *Raspberry Pi*, https://www.raspberrypi.com/documentation/accessories/camera.html. Accessed 11 November 2022.

“Rat in a Maze | Backtracking-2.” *GeeksforGeeks*, 3 August 2022, https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-2/. Accessed 1 December 2022.

“Reading Device for Blind People using Python, OCR and GTTS.” *IJSEA*, https://ijsea.com/archive/volume9/issue4/IJSEA09041003.pdf. Accessed 1 December 2022.

Roeder, David. “What Is the Standard Size of a Pool Table?” *Blatt Billiards*, 23 November 2021, https://blattbilliards.com/blogs/news/what-is-the-standard-size-of-a-pool-table. Accessed 1 December 2022.

Rollins, Leo. “Embedded Communication.” *Electrical and Computer Engineering*, https://users.ece.cmu.edu/~koopman/des_s99/communications/. Accessed 11 November 2022.

Rollins, Leo. “Embedded Communication.” *CMU ECE*, https://users.ece.cmu.edu/~koopman/des_s99/communications/. Accessed 4 December 2022.

Rosebrock, Adrian. “OpenCV Thresholding (cv2.threshold).” *PyImageSearch*, 28 April 2021, https://pyimagesearch.com/2021/04/28/opencv-thresholding-cv2-threshold/. Accessed 11 November 2022.

Sight Machine Inc. “Computer Vision platform using Python.” *SimpleCV*, http://simplecv.org/. Accessed 11 November 2022.

Sinha, Utkarsh. “Circle Hough Transform.” *AI Shack*, https://aishack.in/tutorials/circle-hough-transform/. Accessed 11 November 2022.

“Smart Guidance System for Blind with Wireless Voice Playback.” https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9395973.

Smith, Michael. “PickPocket: A computer billiards shark.” *Artificial Intelligence*, vol. 171, no. 16-17, 2007, pp. 1069-1091. *sciencedirect*, https://www.sciencedirect.com/science/article/pii/S000437020700077X?via%3Dihub.

Smith, Michael. “Running the Table: An AI for Computer Billiards.” *Association for the Advancement of Artificial Intelligence*, AAAI'06: Proceedings of the 21st national conference on Artificial intelligence, 2006, https://www.aaai.org/Papers/AAAI/2006/AAAI06-156.pdf.

“Tek Pal Tactile Low Vision TV Remote Control.” *Maxi Aids*, https://www.maxiaids.com/tek-pal-tactile-low-vision-tv-remote-control. Accessed 1 December 2022.

TensorFlow. “TensorFlow Lite.” *TensorFlow*, Google Inc., 26 May 2022, https://www.tensorflow.org/lite/guide. Accessed 11 November 2022.

TensorFlow. “Why TensorFlow.” *TensorFlow*, Google Inc., https://www.tensorflow.org/about. Accessed 11 November 2022.

“UHF RFID Inlay: AD-172u7 - Avery Dennison.” *Avery Dennison | RFID*, https://rfid.averydennison.com/en/home/product-finder/ad-172u7.html. Accessed 30 November 2022.

“Voice Navigation Based guiding Device for Visually Impaired People.” https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9395981.

Vudrag, Robert. “Choosing the Right Pool Table for Your Home or Business - Quedos.” *Quedos Billiard Tables*, 26 July 2019, https://quedos.com.au/guide-buying-pool-table/. Accessed 1 December 2022.

“What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink - MATLAB & Simulink.” *MathWorks*, https://www.mathworks.com/discovery/slam.html. Accessed 1 December 2022.

“World's smallest Time-of-Flight ranging and gesture detection sensor.” *STMicroelectronics*, 30 May 2016, https://www.st.com/resource/en/datasheet/vl53l0x.pdf. Accessed 2 December 2022.