8-Ball Lifeguard

Authors: Devank Agarwal, Justin Rager, and James Ray

Affiliation: Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— A system designed to help people train their skills at 8-ball pool. The system projects lines onto the pool table to help users align their shots. There is one line showing where to align the cue stick, and lines showing the directions the target and cue ball should go. The cue stick also incorporates sensors to provide feedback to the user after each shot. It contains compasses that allow for the calculation of the angle of the shot, relative to the pool table. There are also accelerometers in the cue stick to track the power**

***Index Terms*— 8-Ball, computer vision, pool, sensor**

I. Introduction

Eight ball pool is a very difficult activity for people to learn. There are not a lot of resources available that are able to help people learn which pool ball to aim for on any given shot, the angle at which they should be shooting, and how to visualize the shot they want to shoot. These are very difficult things for new players to learn on their own. And the easiest way to learn it would be by taking lessons. However, it can become very expensive to receive a high quality pool lesson.

8 Ball Lifeguard is a project that can analyze a game state in pool, provide the best shot available to the user by projecting it onto the table, and help the user line up the shot by using a laser coming out the front of the pool stick and an LED to tell the user when their shot is on the correct angle. This project will help new players tremendously because it takes the most difficult parts of pool off of their hands so they can learn how to correctly hit the cue ball. On top of that, it also helps the user learn to hit the cue ball correctly by helping the user line up the shot and provide feedback to the user on how to improve their shot.

A project similar to this one was done in Spring 2019 but our project differs from their project in 2 major ways. Their project was focused on the game of 9 ball pool while ours is built for 8 ball pool. This is a major difference because in a 9-ball pool you only have one choice of which ball to aim for while in an 8-ball pool you have many different options for which ball to aim at. Another major difference is that our project is incorporating sensors and other features attached to the pool cue. These sensors and other features are designed to help the user be able to line up their shot better as well as provide feedback on how to improve their shot, like should they shoot harder or softer or what angle they were off by for their shot.

II. Use-Case Requirements

In order to make this product something people would want to use, we have defined several use cases for our design. First off, we want our product to be quick. It should be able to read the table position and display the recommended shot quickly, rather than having the user wait for a minute or more before each of their shots. Because of this, we want our design to be able to compute the recommended shot in less than 3 seconds. As this calculation is fairly straightforward for most positions on the table, this should not be difficult. We also want the CV detection for the balls to take less than 5 seconds. Since the balls are all different colors than the table, detecting the positions of all the balls is not complex. The harder part of this process is detecting which of the colored balls are solid color vs striped. This is possible by checking for the amount of white visible to the camera, but since it is more complex, we are giving it a less-strict requirement.

For giving the user feedback, we also want the measured angle and acceleration of the sensors on the pool table to be accurate. For this, we want to make sure the cue is able to update its position quickly and accurately. Our requirement for this is that the pool cue should be able to update its position every 10ms. While we are implementing a system to detect when the cue makes contact with a ball, the cue stick should be able to quickly record this data and send it to the laptop. Being able to quickly record and send the data from the pool cue to the laptop will also be useful for debugging and integration, as it will allow us to quickly run tests instead of waiting for seconds on minutes for the data to update and send. This shows that the requirement is still useful even if we are not constantly polling the stick for its position.

Our design should also not hinder the users. If we make the pool cue much heavier than it previously was, it could alter how they take shots, and when they remove the device, their training will not be transferable. Additionally, if the device weighs too much, it could unbalance the cue stick, which will also make it hard for a user to transition to a regular pool table. To minimize the effect of these, we have imposed a weight limit on the pool cue sensors. All combined, it will weigh less than 0.25lbs, battery included. This should make it that the pool cue is not unbalanced and does not weigh significantly more than a regular cue without the sensors.

Overall, we also want to make sure our device can measurably improve people's skills at the game. Our goal is for people to be able to pocket a ball on nearly every shot while using our device. That may be unfeasible, so we want users to either be able to make shots they couldn't without our

device or significantly improve the accuracy of the shot, even if it still missed. So, in our tests, we want users to either make a shot they missed, or if they missed while using the device, we want an improvement of 5 degrees or more. A 5 degree improvement is significant enough to say it is not random chance, and that using our device helped.

III. Architecture and/or Principle of Operation

Our System includes a pool table with a wooden frame to capture ball positions, a camera, and a projector on top of the wooden frame to process and project said image capturing, and a smart embedded cue stick which shall notify users real time as to how accurate their shot is or isn't. Figure 1 depicts a cartoon version of what the system would look like.

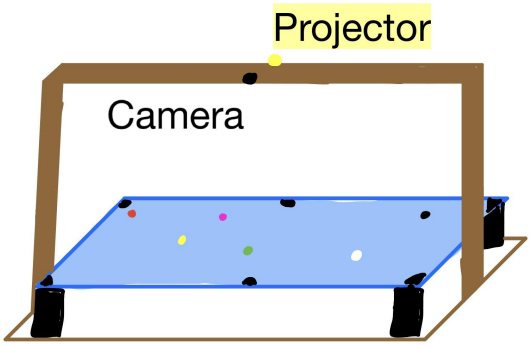


FIGURE 1: A cartoon description of the pool table.

The pool table system has three elements to it apart from the table itself. We have the frame holding the system in place, the camera and the projector. The Frame holding the camera and projector is made up of plywood. The height of the two legs are 3 inches wide and 55 inches tall since the projector needs to be approximately 4 feet away to display the right dimensions onto the table and then we add 7 inches to account for the height of the pool table with the legs. There exists a base on the bottom also made up of plywood, whose dimensions are 42 inches by 18 inches serving as a counterweight. The dimensions were chosen as such to account for the dimensions of the pool table and to add a bit of wiggle room. The top of the frame which would hold the camera and the projector is also 42 inches wide, same as the base, and 3 inches wide, same as the legs.

The camera used by the system is the Logitech C922X. This will sit on top of the frame and will capture live video of the pool table at all given moments. This video stream will be fed to a laptop backend via USB which would handle the Computer Vision aspect of the system and then feed it to the shot calculation. Figure 2 is the block diagram for the backend shot calculation and the AR system.

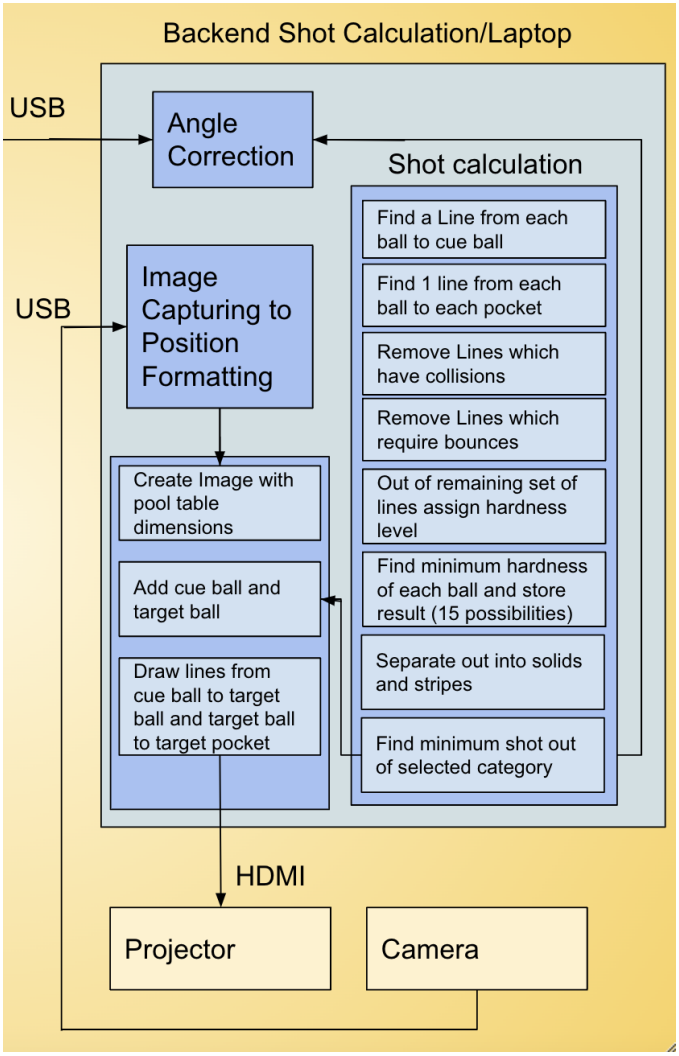


Figure 2:The Block Diagram for the backend shot calculation

The Pool table will also have 4 barcodes which help us correct for lens distortion and camera tilt so the readings are not off. The Backend shot calculation would be responsible for given a grid of balls as x,y coordinates finding the easiest shot for the user to make. This would involve first finding a line to every ball from the cue ball and eliminating balls where a different ball would lie on the same line. Then from the balls make a line to each pocket, remove the lines which would cause a collision with other balls and save the one with the smallest angle delta to the first line from the cue ball. If no lines have been found check for whether or not a shot can be made with a bounce and save that shot. Once the easiest shot for each ball has been calculated, we find the easiest shot from the set of all balls and return that to the image creation pipeline. The image creation pipeline would be responsible for creating the image which the projector would project. Once fed in the shot to make, i.e the target ball and the cue ball along with the lines (in the form of m and c from $y = mx + c$) the system will create an image of the cue ball, the first line from the cue ball to the target ball, the target ball, and the second line from the target ball to the pocket and then display

this image. Once displayed, this image will be used by the projector to display the shot onto the pool table. The projector we are planning on using is the Epson VS330. The image fed to the projector will be sent via an HDMI cable from the backend laptop.

The final subsystem within the project that we would like to talk about is the smart embedded cue stick. Figure 3 is a cartoon depiction of what the cue stick would look like.

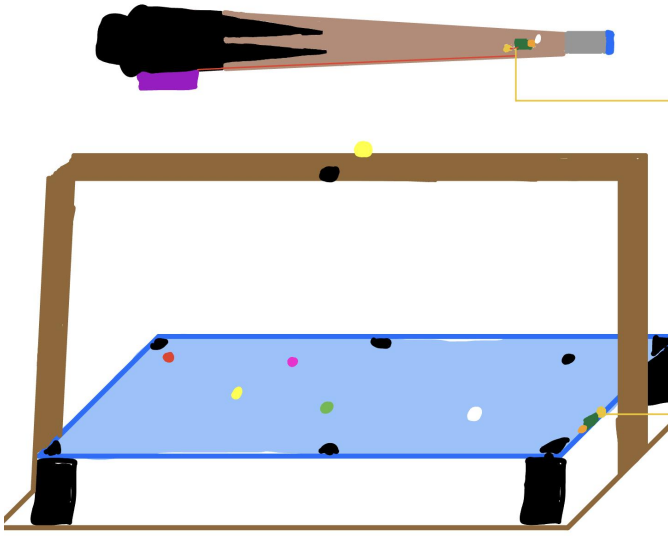


Figure 3: The embedded cue stick and pool table.

There are multiple things involved in the making of the cue stick. The stick itself will have an arduino nano which will act as the microcontroller for communicating with the sensors on the stick as shown in the green in the diagram. The sensor on the stick is an accelerometer and a compass (Adafruit TDK InvenSense ICM-20948 9-DoF IMU (MPU-9250 Upgrade)) which will tell the relative direction the stick is in and the speed at which the shot is being made. These are shown in orange in the diagram. These will feed their values into the microcontroller which will in turn send them via a radio transmitter (Deegoo-FPV 12pcs NRF24L01) to the receiver, the same chip as above to the pool table. These in the diagram are shown in mustard with the line connecting them. The pool table itself will also have an arduino nano and a compass which will help us determine the relative angle the cue stick is at. The receiver will read the values from the cue stick and the arduino will determine the relative angle and the speed of the cue stick and send them to the laptop backend via USB. Once the laptop backend knows the offset of the cue, it will send a signal to the arduino on the pool table which will further send it to the arduino on the cue stick to light up an LED if the angle the cue stick is placed is well aligned. The system on the cue stick is going to be powered by a battery on the stick and the system on the pool table will be powered by a wire from the laptop. Figure 4 shows the block diagram for the entire system.

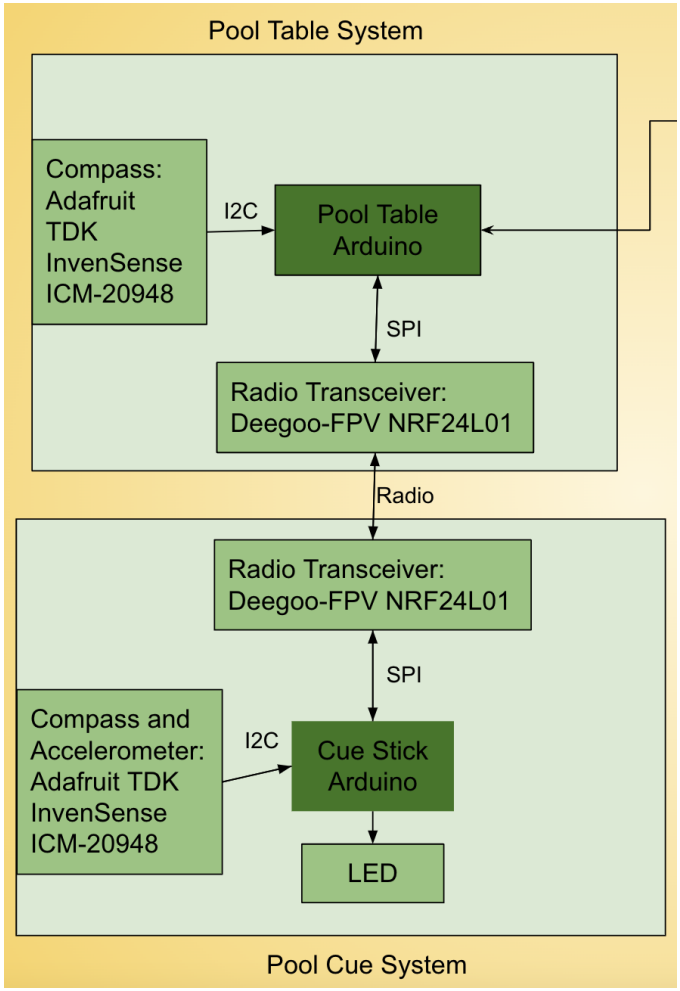


Figure 4: The Block Diagram for the embedded cue stick and table.

The pool table arduino is going to be connected to the laptop backend via USB, the sensor on the stick and the table will communicate with their respective Arduinos via I2C and the arduino will communicate with the radio transceiver via SPI. The transceivers will communicate with each other via radio.

IV. Design Requirements

The design requirements for our smaller implementation for a real time 8-ball pool teaching assistant are based on our use case-requirements. These can be broken down into multiple parts.

A. The Camera Detection System

The first one we shall talk about is the camera detection system. Since the entire system depends upon the initial positions of the balls on the board we are striving to get an estimate of 0.1inch accuracy for the detection of balls. The camera we have chosen is fine grained enough for this and more importantly we plan on correcting any obvious mistakes which the camera may make, these being any distortion caused by the shape of the camera lens and any degree of tilt the system may have due to accidental bumping into the frame. For this we plan on using the NVIDIA Vision

Programming interface. The basic math behind the software is going to be a matrix inverse transformation to get the pixel's actual position to its corrected position. For the angle correction we plan on using 4 barcodes placed on the table, whose offset from a perfect 90 degree angle can be used to correct the tilt. Furthermore, the pool balls must be detected with 95% accuracy.

B. End to End Latency

For our end to end latency we plan on having a 3.1 s end to end latency between shot calculations. We define end to end over here as from the time the balls on the table have 0 movement, that is the game state has been stabilized. The way we plan on achieving this is by having the CV detection have a 10ms latency which is ensured by openCV and then have the shot calculation and image creation take 3 seconds. The way this is being achieved is with parallel programming for each ball where each thread runs its own separate calculation and we have one thread for each ball.

C. Projection and Calculation

For the projection of the new shot created, The system should help improve accuracy of shots by at least 5˚ or less if the improved shot is made. We plan on achieving this by ensuring that the projection and the shot created is not offset by more than 2˚ from the pocket and the projection has a margin of error of less than 0.1 inches. Given a pool table with a 40 inch diagonal length, with these metrics we should be able to achieve this use case requirement.

V. Design Trade Studies

We had to make a few executive decisions in regards to certain tradeoffs. These mainly fall into three different categories, the first of which is the shape and strength of the frame holding the entire system together. The second of these was the hardware we used and the ease associated with them and the third was on how many sources of input we wanted to feed into the system

A. More Rigid or More User Friendly?

For the pool frame, the biggest trade off we had to choose between was the tradeoff between rigidity of the frame to prevent movement of the system and how much of the users view do we block from the pool table. The stronger the frame is the more wood we would need to block the user's view and playing perspective. We chose to lean towards the comfort of playing the game. To counter the rigidity issue, we used a 3 inch width for the wooden frames, and chose to only have the legs on the shorter sides of the table since those have the least shots. To add structural support we added 2 inch by 3 inch by 2 inch cubes on each edge with brackets on the outside. Furthermore we plan on adding diagonal trusses for further support. We made the decision to let the frame wobble by a tiny amount by correcting with software implementations for the tilt in camera angle.

B. Ease of Use or Cheap?

For the hardware, the first issue was whether or not to use RPi or Arduino. For RPi, we would be able to get it for free, but many of the sensor options that we had explored were made for Arduino and would take more work to integrate with the project. In the end, since our budget was not going to be tight, we decided that the extra price for the Arduinos was not an issue. Also, we had to balance the weight of the battery on the cue stick vs the battery life of the sensors. A larger battery life would make it less frustrating for users, but would be more cumbersome when making the shots. In the end, we decided to use a 9V battery due to their still relatively small weight and high availability. Thus, we chose to go with Ease of Use.

C. One source of input or multiple?

We rely on both Computer Vision and sensors instead of simply relying on the Computer Vision module for detection of the cue sticks speed and direction for two reasons. The first of these is that two solely rely on the entire position of the cue stick and its full range of motion. We would need the camera to be placed at a much higher angle than 55 inches from the base which makes the entire system even less rigid and more complex mechanically. Moreover, the Computer Vision module would have to be able to capture frame perfect motion

detection for the cue stick to figure out its speed. This leads into the second reason as to why we chose to have sensors as well.

With both sensors and a Computer Vision system tracking the cue stick, we are able to generate two sources of input to check reliability before sending it to the user. If we only had one of these and the value for the Computer Vision system would be way off then we would have returned faulty values to the user. The compass modules on the cue sticks provide a fairly accurate direction for us to use (accurate to 5 millitagents of measurement) and we are able to back it up with another reliable source. Moreover, to tell the user real time they are on the right angle without having the user have to change their line of view from the cue ball, we needed a signal on the cue stick itself for which we went with an LED. If we didn't make this decision the user would have to line up their shot, turn their vision away while keeping the cue stick perfectly straight which we realized to be hard for a new user to do.

We also can't rely on the sensors by themselves as we would need the Computer Vision system to track the balls. The alternative to this would be to have sensors within the balls themselves however this would require cutting open the balls and changing the weight distribution as well, which we did not have the mechanical engineering skills to do.Thus we decided to go with multiple sources of input.

VI. System Implementation

A. Sensor System on the Pool Table and Cue Stick

The sensor on the pool table serves as both a way to communicate remotely with the cue stick sensors and provides the relative direction of the pool table. This direction is critical for determining where the pool cue is actually pointing. The radio transceiver modules serve as a fast and simple way for the pool table and cue stick to communicate. Over radio, they simply send a bitstream from slave transceivers to a master transceiver. These sensors will communicate with the Arduinos using SPI so as to not interfere with the compass sensor, which communicates using I2C. This microcontroller is also wired into the laptop so it also provides communication between the sensors and the shot computation. The sensors on the pool cue simply serve as ways of collecting data to give the users feedback. The accelerometer also will be able to detect contact with the ball due to a sudden stop, which we can use to time when the shot has been made. The pool cue will also have an LED that turns on when the cue stick is facing the correct direction. To do this, the radio transceivers will need to switch between master and slave in order to transmit data to the cue stick rather than from the cue stick.

B. CV and Shot Projection System

This part of the system is designed such that there will be a camera 4 feet above the table and it will point directly down at the table. From there the camera will pass the images it captures into a python program that is using openCV. This openCV program will look for all of the balls that are on the table. Once it finds all of the balls on the table, it will identify which one is the cue ball as well as which balls are solids vs stripes. It will then pass a graph to the shot calculation system(talked in more detail in the next section) which will contain the (x,y) coordinate of the center of each of the as well as whether each ball is a solid, stripe, the cue ball, or the eight ball.

After the shot calculation system runs, it will then pass an image to the Projection system of the correct shot that should occur. It will then take the image and send it to the projector for the projector to show on the table to the user.

C. Shot Calculation System

The shot calculation will follow a simple algorithm to try to find the easiest shot. It will first receive all of the XY coordinates from the CV system. After it receives the coordinates it will form lines from the cue ball to all of the target balls that have been specified by the user(are they shooting for solids or stripes). It will calculate the slope of the line using the formula $y = mx+b$. Once those lines are created, we will go through all of the lines that have another pool ball in its path and remove them. From the remaining balls that still have lines, we will create 6 possible shots to each of the pockets. Once we create those shots, we will remove any of the shots that would include collisions with either other balls. This second removal of the collisions will ensure that we don't create shots that the path from the cue ball to the target ball is clear but the line from the target ball to the pocket is not clear. After this, if no shot exists we will check for any shots that can exist by bouncing the cue ball and/or target ball off of the wall. From here we will then look at an individual ball to find the easiest shot that exists for that target ball. This will be done by comparing the two slopes of the line from the cue ball to the target ball and the target ball to the pocket and choosing the smallest difference between those two slopes. After we compare for each target ball like we did for each individual ball to find the easiest shot. Once the easiest shot is found, we create the image by adding a line for the cue stick, a line from the cue stick to the target ball, and a line from the target ball to the pocket to an image of the current game state. Finally we send this image to the projector system to project the new image onto the table.

VII. Test, Verification and Validation

For our testing we will be using the 40-in pool table that we purchased. The testing will be done in two phases. The two phases will be broken down into user testing and non user testing. For the user testing, we will recruit 5 newer pool players. We will set up 9 shots for the user to try. These shots will range from easy shot to difficult shots, easy shots being straight line shots while difficult shots necessitating the user to hit the pool ball they are aiming for at an angle to get the ball

into a pocket. These first 9 shots will be done without our system. After the user does the 9 shots, we will then set out the same 9 shots again but this time have the user try the shots with our system. After each shot with and without the system, we will mark down whether the shot is made or not, if the shot missed the angle at which the shot missed, and if any fluke happened in the test that wouldn't make the comparison fair, like if a user made very poor contact with the cue ball. We will then compare, if the user had an increase or decrease in the number of shots and for each shot if there was an improvement on the accuracy of the shot based on the angle. We will consider this test successful for a user if there is an increase in the number of shots that were made or if their accuracy improved by at least 5˚.

While the user test is going on, we also will be watching to see if the frame that we built to hold up the projector and camera. We will ensure that some of the shots the users do are close to where the upright portion of the frame is so that we can analyze how much of an inconvenience the frame is to the user. After the user is done we also will ask them about the frame and if they struggled at all with how to line up the shot and work around the frame.

We also will be conducting separate tests that do not require us to recruit participants. For the other tests, these will be primarily focused on ensuring that the system is accurate and is running fast enough to meet the design and use case requirements. In order to do this we will conduct several different tests which can be broken down into which main system it is testing.

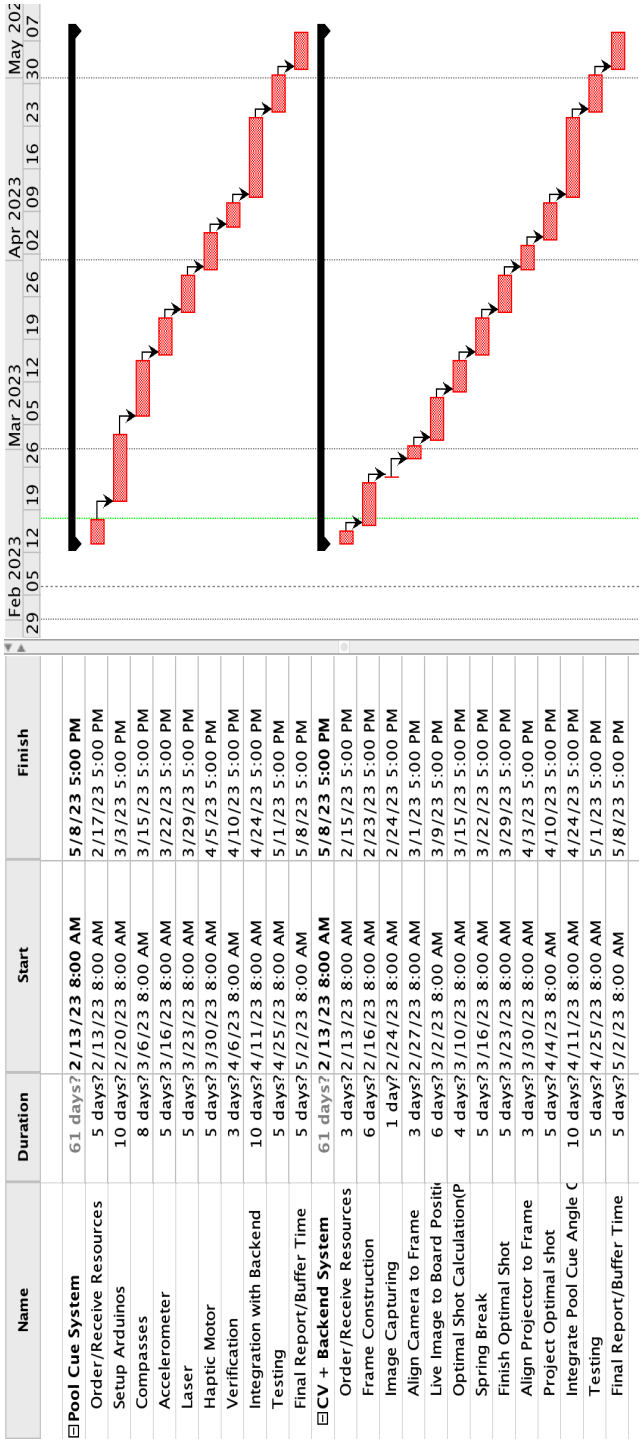
For the pool cue we will conduct three tests. Once we finish designing the pool cue, we will weigh the pool cue with our attachments and compare it to the weight without our attachments to ensure that the weight we added is less than .25 lbs. We also will test how quickly the pool cue updates by changing the angle of the pool cue (i.e. spinning it around) and measuring how long it takes for the angle to be updated in our system. Finally, we will test to ensure that the pool cue's alignment is accurate to 1˚ by setting the pool cue at 3 different angles and measuring the exact angle and comparing that with what our system is outputting.

For the CV system, we will be conducting one test focused on two design requirements. The test will consist of placing 2, 8, and 16 pool balls on the table in various positions. The positions will test edge cases where balls are all right next to each other or all along the sides of the pool table. From here we will analyze the CV results to ensure that pool balls are being detected on the table with at least a 95% accuracy rate. This will be done by adding how many balls were detected by the CV and comparing it to the true number of balls that are on the pool table. We also will be analyzing the positions that the CV says the balls are on the table compared to their true position. We will ensure that the position of the balls are within .1 in of their actual position on the table so that the shot calculations are all accurate.

Finally we will be testing the shot calculation and projection system by giving it different placements of the balls on the pool table. From there we will time how long it takes our system to calculate the best shot that the user should do. According to the requirements, this calculation should take no longer than 3 seconds. From there, we will analyze the shot calculation and the angle that is calculated for the shot. For the final shot that our algorithm chooses, it should be off by no more than 2˚. Finally we will then project the image onto the table and analyze how accurate the projected image is to the table and the correct shot. This image needs to be within .1 in of the actual position on the table for this test to be successful.

VIII. Project Management

A. Schedule



Name	Duration	Start	Finish
Pool Cue System	61 days?	**2/13/23 8:00 AM**	**5/8/23 5:00 PM**
Order/Receive Resources	5 days?	2/13/23 8:00 AM	2/17/23 5:00 PM
Setup Arduinos	10 days?	2/20/23 8:00 AM	3/3/23 5:00 PM
Compasses	8 days?	3/6/23 8:00 AM	3/15/23 5:00 PM
Accelerometer	5 days?	3/16/23 8:00 AM	3/22/23 5:00 PM
Laser	5 days?	3/23/23 8:00 AM	3/29/23 5:00 PM
Haptic Motor	5 days?	3/30/23 8:00 AM	4/5/23 5:00 PM
Verification	3 days?	4/6/23 8:00 AM	4/10/23 5:00 PM
Integration with Backend	10 days?	4/11/23 8:00 AM	4/24/23 5:00 PM
Testing	5 days?	4/25/23 8:00 AM	5/1/23 5:00 PM
Final Report/Buffer Time	5 days?	5/2/23 8:00 AM	5/8/23 5:00 PM
CV + Backend System	61 days?	**2/13/23 8:00 AM**	**5/8/23 5:00 PM**
Order/Receive Resources	3 days?	2/13/23 8:00 AM	2/15/23 5:00 PM
Frame Construction	6 days?	2/16/23 8:00 AM	2/23/23 5:00 PM
Image Capturing	1 day?	2/24/23 8:00 AM	2/24/23 5:00 PM
Align Camera to Frame	3 days?	2/27/23 8:00 AM	3/1/23 5:00 PM
Live Image to Board Positi	6 days?	3/2/23 8:00 AM	3/9/23 5:00 PM
Optimal Shot Calculation(P	4 days?	3/10/23 8:00 AM	3/15/23 5:00 PM
Spring Break	5 days?	3/16/23 8:00 AM	3/22/23 5:00 PM
Finish Optimal Shot	5 days?	3/23/23 8:00 AM	3/29/23 5:00 PM
Align Projector to Frame	3 days?	3/30/23 8:00 AM	4/3/23 5:00 PM
Project Optimal shot	5 days?	4/4/23 8:00 AM	4/10/23 5:00 PM
Integrate Pool Cue Angle C	10 days?	4/11/23 8:00 AM	4/24/23 5:00 PM
Testing	5 days?	4/25/23 8:00 AM	5/1/23 5:00 PM
Final Report/Buffer Time	5 days?	5/2/23 8:00 AM	5/8/23 5:00 PM

Fig. 1. Schedule example with milestones and team responsibilities

The above figure is our schedule. The bolded rows represent the two major sections of our project. The unbolded rows each represent the milestones of each section. The duration is also only counting weekdays, so for example, a duration of 5 days represents one week.

B. Team Member Responsibilities

Devank's primary responsibility is to create the shot calculation equation the laptop will use to compute the recommended shot. To do this, he first needs to create a heuristic on what quantitative values can be used to find the "best" shot. He also is the main person responsible for creating the frame that will support the camera and projector above the pool table.

Justin's primary responsibility is to arrange and calibrate the sensors that will go on the cue stick. He is also responsible for the communication between the microcontrollers on the cue stick and table, as well as communication between the laptop and microcontrollers.

Jimmy's primary responsibility is to create and train the CV model. He also is responsible for helping construct the frame for the pool table. He also is supposed to help with the integration of the projector and camera model to the laptop.

C. Bill of Materials and Budget

The bill of materials is shown at the end of the document. Included in the figure is the price of each item. The total price of all the items is listed in the bottom row. The items that are listed as $0 are items that we were able to acquire for free. Additionally, the third microcontroller is a spare and not necessary.

D. Risk Mitigation Plans

For this project, we need to successfully integrate all of our components together. We have never worked with CV in depth before, so learning how to train the model is new territory for our group. This is lessened by being able to base our model off of existing models, so we simply need to adjust already known models to fit our use case. We also have not done work in microcontroller-to-microcontroller communication before, which is important to the project. However as this is not an unknown area, there are plenty of guides to help us make sure everything works correctly. Another risk is that the sensors on the cue stick may not be able to transfer data quickly enough to the laptop. The sensors chosen are able to transmit data at a high enough bandwidth that this should not be an issue.

IX. Related Work

There have been a couple similar projects to what we have done. The main work that we are building off of is an ECE capstone project from the spring semester of 2019. It was by Team B9: Breaktime. Their work was similar to our project. They built a CV system that would track all of the balls in a game of 9 ball pool and help line up the next shot for the user. We are using this system as a jumping off point for our project. We are using the projector and camera that the other group used for their project. This has allowed us to simplify

the beginning steps of finding what products we should use in our project and instead move on to the more difficult portions of our project.

Another work that our project is similar to is a youtube video by Stuff Made Here. This youtube video demonstrates a robotic pool cue that when put next to the cue ball, it would calculate the next shot that should happen and automatically shoot it for the user without the user doing anything, The difference with our projects is that ours is designed to help the user how to play while their system is built to play for the user.

X. Summary

Our project is a learning tool to help people become better at 8-ball. It will help users learn about how hard to hit the pool ball, as well as what angle they need to hit it at in order to pocket a ball. To help users see the correct angle to hit the ball, we are projecting lines showing the direction the pool cue should hit the pool ball, the direction the pool ball should go after the shot, and what direction the target ball should move once the pool ball hits it. We also will have a light on the pool cue that will light up once the pool cue is in the correct direction. This will provide a secondary guide for the user and serves as a backup if the projector breaks. While there is no good way to indicate the exact level of force the user needs to use when shooting, we are able to provide feedback about the strength they used and if it was stronger or weaker than our calculated value. To do this, we have an accelerometer on the cue stick to measure the force of the impact and will display it on the laptop screen. One challenge for implementation is deciding where the sensors will be on the cue stick. The closer they are to the tip of the cue, the better data they are able to collect. However, that also increases the likelihood that the sensors will interfere with the users while taking shots.

Glossary of Acronyms

CV – Computer vision
HTML - Hypertext Markup Language
I2C – Inter-integrated circuit
LED - Light-emitting Diode
ML – Machine learning
RPi – Raspberry Pi
SPI – Serial Peripheral Interface
USB - Universal Serial Bus

References

[1] "Automatic pool stick vs. strangers" *YouTube,* uploaded by Stuff Made Here, 15 February 2021, https://youtu.be/vsTTXYxydOE

[2] Lambert, Graham. "Wireless Communication between Two Arduinos." *Circuit Basics*, Circuit Basics, 17 Nov. 2021, https://www.circuitbasics.com/wireless-communication-between-two-arduinos/

[3] Ou, Xu, and Kim. *BreakTime Augmented Reality Pool Guidance System* 2019

[4] Siepert, Bryan. "Adafruit TDK Invensense ICM-20948 9-Dof Imu." *Adafruit Learning System*, Adafruit, 5 Aug. 2020, https://learn.adafruit.com/adafruit-tdk-invensense-icm-20948-9-dof-imu

[5] "VPI - Vision Programming Interface: Lens Distortion Correction." Docs.nvidia.com, docs.nvidia.com/vpi/algo_ldc.html.

Item	Price	Part Name	Manufacturer
pool table	$113.46	Hathaway Breakout 40-in Tabletop Pool Table	Hathaway
RF transceivers	$15.15	Deegoo-FPV 12pcs NRF24L01	Deegoo
Accelerometer + compass x4	$75.07	Adafruit TDK InvenSense ICM-20948 9-DoF IMU (MPU-9250 Upgrade)	Adafruit
Microcontroller x3	$81.62	Arduino Nano	Arduino
Wood	$50	Plywood	Home Depot
Laptop	$0	Macbook Pro 13in	Apple
Camera	$0	Logitech C922X	Logitech
Projector	$0	Epson VS330	Epson
Light	$0	LED	unknown
total	$335.30		