C0: CueTips

Andrew Gao, Debrina Angelica, and Tjun Jet Ong

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— This is an eight-ball pool training system for beginners and professionals alike by helping players build intuition about the trajectory of their hits. Our system processes the state of the pool table and displays predicted trajectories in real-time to the user as they move their cue stick. Currently, the state of the art are attachments to the cue stick for assisting aim or controlling the player's force. Our system allows one to simulate a real game of pool as closely as possible with minimal augmentation to the cue stick. Furthermore, our system offers far better training by displaying all possible trajectories.**

***Index Terms*—billiards, eight-ball, pool, trajectory prediction, computer vision**

I. Introduction

Our project, CueTips, is an assistive system that helps beginner and advanced Eight-Ball Pool players improve their skills. Eight ball pool is a difficult game to learn, primarily due to the high degree of accuracy needed to make a successful shot to pocket a ball. Very slight differences in the angle of the player's aim can have a profound impact on the trajectory of the ball they hit. Without expert guidance to correct players' aim, it can take a very long time to develop an intuition for where the ball will go depending on the player's aim. While it is possible for players to hire coaches to improve their aim, doing so would not be feasible for the average player who wants to play recreationally.

Our product is an assistive feedback system that provides real-time predicted trajectories based on the player's cue stick position. As the player moves their cue stick around, they will be able to see a projection line showing them the direction and angle in which the ball they are aiming at will go. This will help players know whether they will be able to pocket the ball and subsequently give them insight on the correct aim location.

A competition of our project is a similar project that was done in the Spring 2023 semester for Carnegie Mellon University's (CMU) Electrical and Computer Engineering (ECE) Capstone course. This project gave players a single suggestion on which ball to hit, and from which angle, based on the state of the balls on the pool table. The advantage of our project compared to this one is that our project will provide real-time feedback to users, which will better allow users to improve their intuition for aiming. If the user's aim is off, they will be able to notice that from our trajectory feedback, which will allow them to correct themselves. This is more beneficial in helping users learn as it provides more personalized feedback based on the user's movements of the cue stick. Furthermore, our system gives users the flexibility to choose which ball to hit, which would allow them to improve their decision-making skills in a game of pool as they get to observe the trajectories of the different balls on the table. Our physics model also has some features that differ from the prior project. Firstly, our system will allow users to practice bank shots since we will also display the trajectory of the cue ball if it is aimed towards the walls of the table. In addition, our physics model will not only show the trajectory of the target ball, but also show the trajectory of the cue ball's deflection due to its collision with the target ball. This allows users to get an idea of where the cue ball will land when it comes back to rest.

II. Use-Case Requirements

The primary goal of our project is to serve as a tool that effectively helps users learn to play pool.

One of the features required to facilitate fast learning is to ensure that our system's performance is fast. We want our system to be able to react instantaneously to user movement as they play the game. Hence, we must be able to achieve a latency of at most 100ms. This latency is measured from the time the user shifts their position to the time that our system provides and updates a projection for the trajectory of the user's aim. We chose a threshold of 100ms as research has shown that this duration is the threshold required to create the illusion of an instantaneous response [1]. This immediate feedback is not only important in creating a responsive user interface, but also crucial in helping users learn instantly from their mistakes if their aim is not accurate.

Accuracy is another crucial use case requirement for our project. In order to effectively help our users learn how to improve their aim when playing pool, we must be able to provide them with accurate predictions on the ball trajectory based on their cue stick's position. We aim to have our predictions accurate with at most 2 degrees of error. 2 degrees error is the measured angle between the line of our predicted trajectory, and the line of the ball's actual path after it collides with the cue ball. The reason we select an error of 2 degrees is because the pockets are of a width that would allow a ball to still fall in even if it were 2 degrees to the left or right of its predicted trajectory [2]. This is assuming that the trajectory of the ball was directed to the center of the pocket.

To facilitate the high accuracy our model aims for, we must also have accurate object detection models. We aim to be able

to detect ball position within 0.2 inches of their actual position. If our model's perceived location of a ball skews too far away from the ball's true position, this would cause inaccuracies in our physics calculations that drive the trajectory prediction.

CueTips addresses some crucial aspects of public health, safety, and welfare by promoting mental well-being, physical activity, and cognitive health. Our product aims to alleviate the stress associated with the learning process of pool, fostering a positive and supportive environment for individuals to learn pool. Furthermore, the interactive nature of the game serves as a form of recreational exercise, allowing individuals to move away from their sedentary lifestyles. In addition, it could also be useful for older individuals to engage in activities that stimulate cognitive functions and maintain mental acuity.

In terms of social factors, our product aims to be accessible and reach a wide range of users. This motivates our choice of outputting visual trajectory predictions, as it would inherently be community-building by serving as a massive social interest in many parts of the world. By being accessible to everyone, CueTips creates room for substantial economic potential. We could strategically partner with entertainment venues, gaming centers, and sports bars to target the demographic who are not only passionate about the game of pool, but also seeking interactive and technologically advanced gaming experiences. Through upfront hardware sales, subscription models, and potential collaborations with game developers for exclusive content, there is a lot of potential for our product to contribute significantly to the economy.

III. Architecture and/or Principle of Operation

Our project consists of several physical components. It consists of a pool table mounted onto a shelf to house a camera for object detection and a projector to project recommended trajectories, an inertial measurement unit integrated into a cue stick to collect user cue stick data on position and orientation, and a web application to display the user's shot and provide personalized recommendations.

A. Physical Structure

The physical structure of our set-up consists of a RayChee pool table mounted onto a rack. The pool table is directly mounted onto one of the racks of the frame. This structure is 48 inches wide, 24 inches long, and 72 inches tall. There will also be a second rack that is 48 inches above the pool table. After some testing and calibration, we determined that the optimal distance from the projector and the camera to achieve the best field of view is when they are both placed 44 inches above the pool table. We raised that by four inches in order to ensure some additional leeway in case we needed a greater field of view when performing our actual detection model.

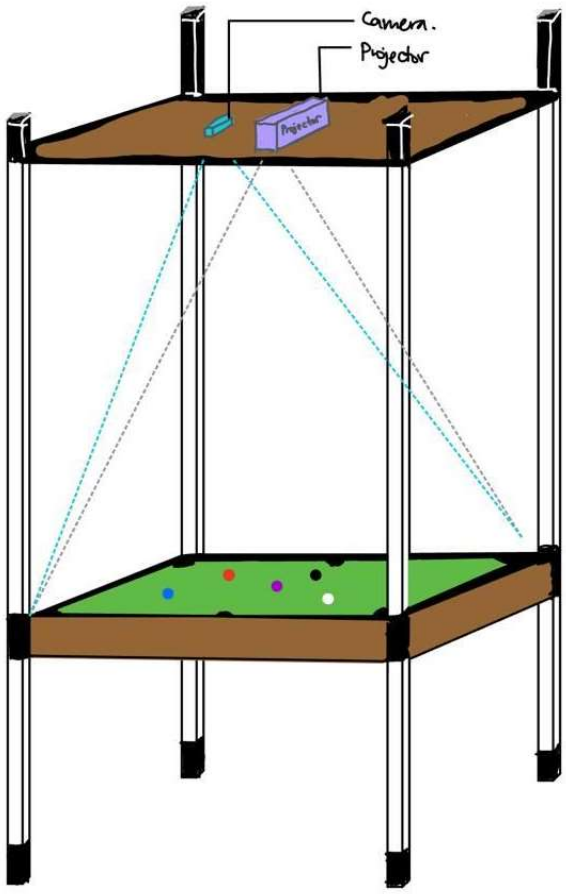


Fig. 1. Pool table incorporated into a shelving unit, mounted on a rack

B. Camera and Computer Vision

The camera we are using to perform the computer vision model is the Logitech C922 Pro Stream 1080p webcam. We will cut out the appropriate portion of our plywood rack and mount the camera nicely into the cut-out, in order to achieve a good image capture of the table. The camera will be connected directly to an embedded system that we will use for this project, the NVIDIA Jetson Nano via a USB Cable. The camera gives a 78-degree Field of View, meaning that if we want to get a full capture of the pool table, we will at least have to mount it 33 inches above the pool table.

The NVIDIA Jetson Nano will run the backend computer vision model subsystem. This will include the detection of the current state of balls on the table, detecting which ball is the cue ball, the cue stick, pockets, and walls. This information will then be used as input to our physics model, which we will use to predict the trajectory of the shot the user will take. Fig. 2 shows the full process of what happens from the moment the camera inputs are detected, to the trajectory calculations, and finally to the projection of the predicted trajectory.

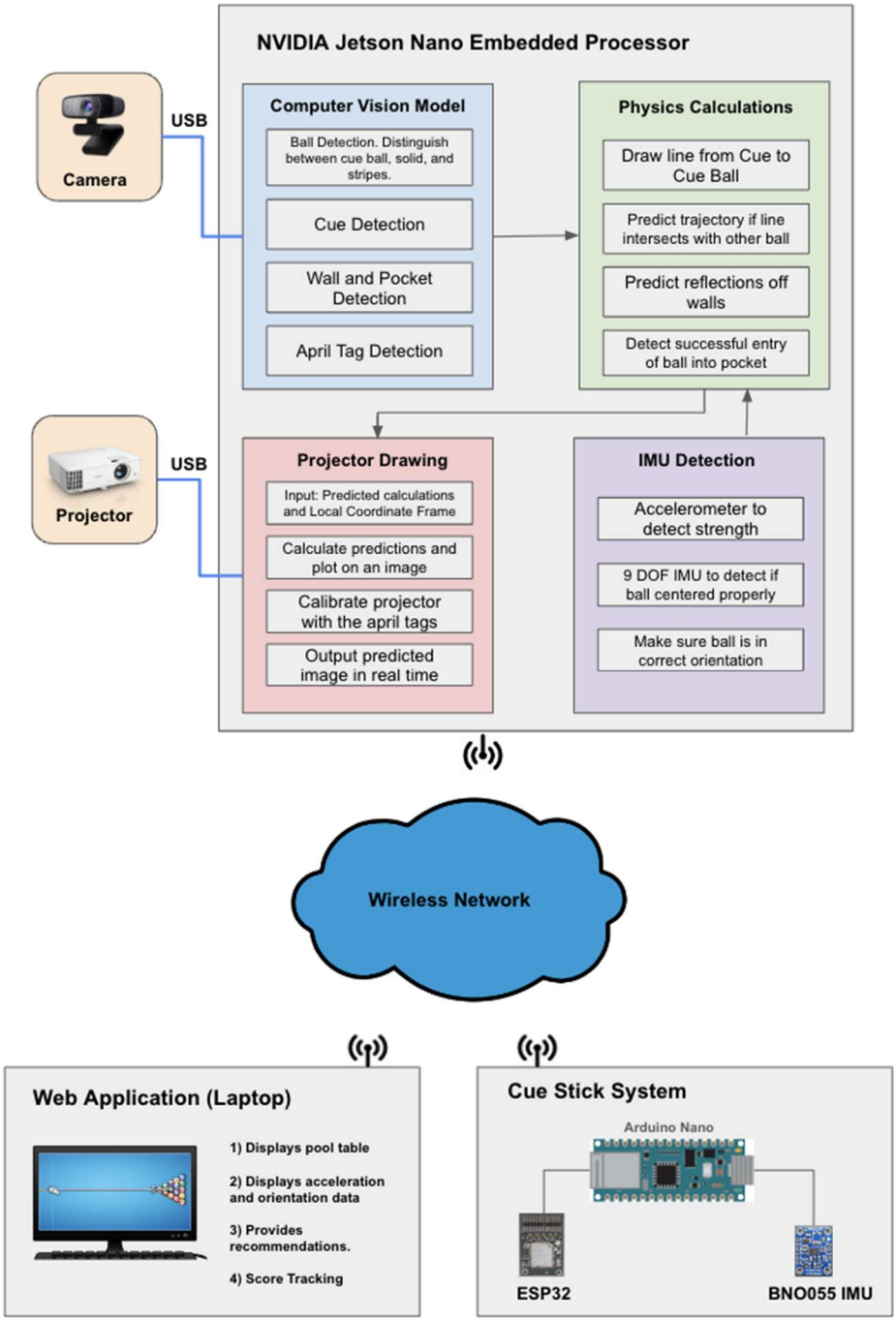


Fig. 2. Block diagram of full network architecture

C. Physics Model

There are five key outputs of the computer vision model that we will use: An array that contains the coordinates of the centers and radius of all colored balls (segregated into solids and stripes), the center and radius of the cue ball, the location of the cue stick, an array of line equations of the four walls, and an array that contains the coordinates of the six pockets. Our physics subsystem contains two main parts, a function to calculate reflections against the walls, and a subsystem that calculates the predicted trajectory.

In order to implement the wall reflections and the trajectory predictions, we first identify the locations of the cue ball's center and two points on the cue stick. Whenever a user aims at a cue ball, these three points should be collinear. We will then extrapolate a line across these three points to estimate the trajectory that the cue ball will go. As we extend this line further, we will meet either one of the three cases: There is no obstacle, there is a wall that intersects the line, or there is a ball that intersects the line. After taking into account all of these, we will output the final predicted trajectory that the cue ball will

move in as well as where the balls they collide with will end up in. This predicted trajectory line will be sent as an input to our projector subsystem. This entire subsystem will also be run on the NVIDIA Jetson Nano.

D. Projector

The projector we are using for this project is the VOPLLS 1080P Full HD Supported Video Projector. This projector takes in the coordinates of the predicted trajectory line and plots it as a white line on a black background. This system will also make sure that if the line coincides with any of the pockets, it will stop the trajectory of the line to indicate that the ball will go in. The projector will also be connected to the NVIDIA Jetson Nano via HDMI Cable.

E. Cue Stick

We will also mount a system onto the cue stick in order to obtain acceleration, magnetic orientation, and angular velocity. The proposed design of this system is depicted in Fig. 3. There will be three pieces of hardware mounted onto the cue stick. In order to sense the acceleration, magnetic orientation, and angular velocity, we will use a Bosch BNO055 9-Degree of Freedom (DoF) Inertial Measurement Unit (IMU). We will be using the ESP-WROOM-32 Development Board as our Microcontroller, which also has the ability to support 2.4GHz Dual-Mode Wi-Fi and Bluetooth. This is compatible with the Arduino IDE, which will allow us to interface with our IMU, and allows us to communicate with the NVIDIA Jetson Nano via Wi-Fi. The purpose of this is to eventually provide recommendations to the pool player to improve their shot. We will also play back the user's shot and indicate the velocity and spin of the user's shot.

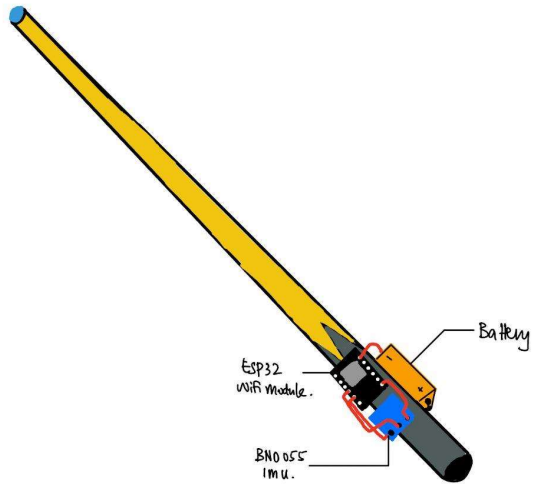


Fig. 3. System mounted onto the cue stick in order to obtain acceleration, magnetic orientation, and angular velocity

F. Web Application

Our web application provides a nice user interface to livestream the video output and displays the ball trajectories onto the table. The application will also display essential parameters such as user-applied force, the orientation of the cue stick, and mild spin dynamics depending on where the ball ended up. A distinctive feature of this web application lies in the provision of actionable feedback, promptly communicating errors to users and offering strategic insights for improvement based on individual playstyles. We will use React to create our front-end web application. For the backend, we will deploy a Flask server to receive video input from the NVIDIA Jetson Nano, and de-serialize the input from the IMU.

IV. Design Requirements

Our design requirements for CueTips are a translation of our use case requirements defined in section II.

A. Computer Vision Subsystem

The CV subsystem must be able to accurately detect the objects on the pool table. We aim for a maximum ball detection error of no more than 0.2 inches, which is measured from the center of the ball detection to the center of the ball's actual location. Furthermore, we aim to detect the balls on the table 95% of the time. To ensure as accurate readings as possible, we plan to perform predictions on 10 frames and correlate between them to figure out where the balls are. This is crucial as this data is used by our physics model to calculate the trajectories of a ball depending on the cue stick's aim. Furthermore, we must also ensure that our game environment has constant visual and lighting conditions, to ensure the most consistent detections. To achieve this, we will project the ball detections, and cue stick detections onto the table, and physically measure the errors between the detection and actual.

B. Computer Vision Subsystem

We aim to provide feedback to the user with a latency of at most 100ms. This latency is measured from the time the state of the game changes (the user repositions the cue stick) to the time that our system outputs a projection onto the table. In order to achieve this, we plan to pipeline our system to maximize the number of frames we can process at a time. We will parallelize the different phases in our system — retrieving data from the camera, performing object detection on the frames, and executing physics calculations on the processed frames. This means that while the physics model is performing its calculations on a certain frame, the CV model is already performing object detections on the next frame. Furthermore, we plan to make use of existing NumPy vectorization techniques for more efficient computation on NumPy arrays and data sequences.

C. Projection subsystem

The projector subsystem is created so that there is an intuitive user interface so that it is easy for the user to follow the feedback provided by our system. We will use a projector

mounted overhead to display the ball trajectories predicted by our model onto the pool table. This allows the users to clearly see the trajectories as lines which they can then use to inform them of whether a ball will successfully go into the pocket. We must also ensure that the output trajectory is no more than 2 degrees from the trajectory executed by the user in real life. This means that the actual path the target ball follows after it is hit should be no more than 2 degrees to the left or right of the predicted trajectory.

V. Design Trade Studies

In our system, we made four primary tradeoff decisions: two software-related tradeoffs, and two hardware-related tradeoffs. We discuss each in more detail below.

A. Power vs Efficiency for Latency Requirements

Our first large hardware-related tradeoff is deciding on what hardware the computer vision model and physics engine would be run on. Originally, we wanted to run all our software systems on an FPGA for efficiency, as well as solving a challenging optimization problem. However, we quickly realized that running the bulk of all our software on an FPGA deviates from the actual purpose of the project; it is fundamentally a technical flair that does not add additional functionality. It added no real value to our use case requirements and the product itself, and from a real-world perspective it makes more sense for users to be able to use their own computers to be able to run the software. We decided a few weeks into the project that running our software on an FPGA would be a nice-to-have, but it pulled a lot of effort away from writing the actual software that would do video processing, physics computations. Further, after some initial testing with OpenCV on our laptops, we realized that putting this software on an FPGA would fail to meet the latency requirement. Running simple algorithms like edge and contour detection took up ~20ms on Apple's M1 MacBook Pro, and FPGAs run at orders of magnitude slower clock speed. We did not want to put ourselves in a situation trying to optimize every line of code in order to meet the latency requirement if developing on an FPGA. Furthermore, there were a number of advantages to developing our software in Python versus C/C++, which we would have used if developing on the FPGA. Our team has far more experience working with Python than C/C++; using C/C++ would have resulted in more wasted time familiarizing ourselves with the language and its subtle intricacies. Additionally, Python has much better support in terms of open source libraries to build upon. It also allows us to test and develop software much faster due to its ease of use. For all the reasons above, we chose to use a laptop to run our computer vision, physics engine, and trajectory prediction software.

B. Camera and projector mount construction

One of the main issues that we and other students in the past have run into building a pool-based project is constructing a rigid mount for a camera/projector. This was a well-documented issue with a past ECE capstone project - "8-ball lifeguard". Their group's solution was to build a wooden, LED-studded mount attached to the pool table to hold both their projector and camera. After discussion with both this group and other capstone faculty, we learned that they had significant issues constructing the mount and having the camera and projector remain still. After seeing the issues multiple other groups encountered, we decided to sidestep construction of the mount altogether and opted to buy the mount. We settled on a large, metal shelf for the mount which perfectly fit around the pool table. This allowed us to attach both the camera and projector to the top with minimal extra effort. The crucial tradeoff we made here was using a significant amount of our budget on the shelf versus spending time to construct a custom solution. We opted for this approach because it seemed that for past teams, multiple weeks were spent trying to get the mount to work, time which could be better spent on other areas of the project

C. Zero machine learning for object detection

The first large software tradeoff we made was opting for zero machine learning in our computer vision/object detection system. This decision was primarily made keeping in mind the latency use case requirement set having end-to-end prediction within 100ms. Our group did some initial testing with multiple machine learning-based object detection models (Region-Based Convolutional Neural Network (R-CNN), YOLO, etc.) and measured the time it took for each frame to be processed. We realized that these machine learning models were at least an order of magnitude slower than *without*. Thus, in order to meet our latency requirement, we abandoned using machine learning - at least for the computer vision/object detection aspect of this project. The tradeoff was that developing the object detection system became a lot more complex. Machine learning models like YOLO for object detection are very straightforward to use and are full, out-of-the-box solutions. However, opting for a non-machine learning approach meant that we had to manually detect cue balls, sticks, and use clever ways to differentiate between objects (e.g. color, shape, size). This also means that our environment must be well-lit and as consistent as possible.

D. Building the physics engine

The other large software tradeoff we made was concerning the physics engine. From the beginning, our group tried to build the physics engine by extending existing work, papers, and other open-source libraries. However, after trying multiple open-source libraries and attempting to adapt various pool game engines to our use case, we were faced with a decision: keep attempting to adapt existing work or build our own engine from scratch. After a few weeks of unsuccessfully trying to adapt existing work, we built our own physics engine and prediction systems from scratch. The main reason for this decision was because many existing pool physics engines were used for pool video games or were far too complicated for our needs. The bulk of our physics engine lines in the trajectory prediction, which is more geometry heavy. Most of the physics engines currently out there take into account other factors like mass, coefficient of friction, spin, among other things. As such, they were unnecessarily cumbersome, and it took more effort to

integrate an existing physics engine into our system compared to writing one from scratch. It took us around 1-2 weeks to write a custom physics engine from scratch, far less time than opting for a third-party solution.

VI. System Implementation

The trajectory projection system consists of the following systems: computer vision system, cue stick system, physics engine, projector system, and web application.

A. Computer Vision System

For our computer vision system, we relied heavily on the OpenCV library and utilized zero machine learning. The computer vision system consists of five components: ball detection, cue ball detection, pocket detection, wall detection, and cue stick detection. Each of these components is modularized as a Python class for ease-of-use when integrating all together. The ball detection module (class BallDetection) is initialized with the filename, which tells the module where to look for to load the frame in. The most important function in the BallDetection class is getBalls(). This loads the image from the specified file path and runs cv2.HoughCircles on the image; HoughCircles is run with a minimum and maximum radius of the desired object. This value was calculated by hand based on the actual radius of the balls as seen in each frame from the mounted camera. Then, if any circles are detected, they are drawn onto the original image. This image is then outputted from the function. For cue ball detection, we differentiate it specially from the other cue balls based on color. After multiple tests, we found a range of acceptable RGB values that identifies the cue ball deterministically. The pocket detection module (class PocketDetection) is structured similarly to the BallDetection class. It requires the file path of the image to be processed and has a similarly named getPockets function which is the centerpiece of the class. Again, the getPockets() function runs HoughCircles with custom parameters for the minimum and maximum radius of the pockets. By specifying the range of radii for both balls and pockets, we are able to differentiate between the two objects despite them both being circular from top-down. Lastly, there is the wall detection module (class WallDetection) which is responsible for identifying the pool table walls. It is similarly structured to the other two modules, and the main function used is getWalls(). First, the image is loaded in via a provided file path to the class. It is run through a combination of edge detection (cv2.Canny) and contour detection (cv2.findContours), and these contours are the overlay onto a black background with the same dimensions. Then, this is run through cv2.HoughLines to obtain a list of the lines detected. To classify the lines as either the top or bottom walls versus the left or right walls of the pool table, some clever heuristics are used. Since the pool table is in the shape of a rectangle, the left/right walls should be closer to the center of the image (perpendicular distance) compared to the top/bottom walls. Additionally, the angles of the walls should be either 90 degrees or 0 degrees as well (normalized). These two heuristics are combined to classify the wall as horizontal or vertical lines. The four lines are then converted from polar coordinates, the format HoughLines returns, into rectangular. Lastly, for the cue stick detection module (class CueStickDetection), this draws primarily upon contour detection to identify the cue stick. By making sure we identify the balls, wall edges, and pockets correctly, we are able to disqualify these objects when running contour detection on the frame. The main function in this module is getCueStick() which returns the coordinates of roughly the tip of the cue stick. This system purely consists of software components.

B. Cue Stick & IMU System

The cue stick/IMU system consists of both hardware and software components. For hardware components, the cue stick has an Adafruit BNO055 9DOF IMU and an ESP-WROOM-32 2.4 GHz Wi-Fi microcontroller processor attached to it. The data we are primarily concerned with is both the gyroscopic data (roll, pitch, heave) and the accelerometer data. The ESP-32 is programmed with Arduino code and polls data from the IMU. We use the getVector function to fetch data from both VECTOR_GYROSCOPE and VECTOR_ACCELEROMETER. The VECTOR_GYROSCOPE consists of three floats representing roll, pitch, and heave respectively in units of degrees. The VECTOR_ACCELEROMETER is just the acceleration measured in m/s^2. To send this wirelessly to the Jetson Nano running the physics engine, computer vision system, we initialize a WIFI Server on port 80 and connect. We then write bytes to the connected client on the other end (which is the CV, physics engine system) in the format of an HTTP request sending this data.

C. Physics Engine

The physics engine was written using OpenCV and NumPy, and is implemented as a Python class (class Physics). The key functions in our physics engine are find_new_point_on_line, calculate_circle_and_cue, calculate_trajectory. find_new_point_on_line computes the point between two balls where a collision occurs. We determined that a collision only occurs if the distance between the centers of the cue ball and other ball is roughly ~2R (R is radius of one of the pool balls, we assume them identical). The trajectory of the cue ball is represented as a line, and the other balls on the pool table are passed in as arguments. Each ball is checked against the cue ball trajectory to check for collisions, and if there is not a collision, we move on. If a collision occurs, however, then we solve equations to find two possible intersection points by solving a quadratic. The closest point is taken. For calculate_circle_and_cue, this function is responsible for identifying the cue stick and balls on the pool table. It calls cv2.minEnclosingCircle to find circles that represent the balls and distinguishes the cue stick based on its radius size. This function is vital in representing the state of the pool table as concrete data in the form of two dictionaries: one containing data for the cue stick, and another for the centers and radii for the balls on the table. The last function, calculate_trajectory, brings the other functions together. Given a pool table frame, it calls calculate_circle_and_cue to process the image into a more code-friendly format (Python dictionary). Then, find_new_point_on_line is called to find the collision point (if

any) between the cue ball and some other ball. We take this data and do some mathematical computation to figure out the resultant vectors of the cue ball as well as the ball it collides with. These vectors are then returned for the projector module to display.

D. Projector System

The projector system is the final subsystem in the trajectory prediction system. It takes in the original frame processed as well as a Python dictionary containing a list of vectors to display. These vectors are the predicted trajectories after the cue ball collides with the target. It first creates an equal-dimension black background (calling np.zeros_like), then loops through the list of vectors and plots them as white lines upon the black background. The maximum contrast is a decision we made to ensure that the lines are shown on the pool table even in well-lit environments.

E. Web application

The web application currently allows the user to self-calibrate their strike with respect to both force and orientation of the pool stick, helping them hold it correctly. The web application is built with React and Flask (Python). A small Flask server is set up using SqLite for a small, lightweight database and exposes two endpoints "/accel", "/gyro" that accept POST requests, and an endpoint "/update" that accepts GET requests. The server receives both accelerometer and gyroscope data on the /accel and /gyro endpoints respectively. The frontend polls the /update endpoint every ~200ms and receives the gyroscope and accelerometer data captured in that time period. It then displays to the user whether they are holding the stick correctly, and how much force was being applied to the cue stick. Because the poll time is frequent, the updates the user sees on the web application appear near instantaneous. The gyroscope data is also displayed back to the user (roll, pitch, and heave), and if the stick is not level (primarily concerned with pitch), then it will guide the user with directional arrows on how to reposition the cue stick.

F. Public health, Safety, Welfare Considerations

In our system implementation, we considered a number of public health, safety, and welfare considerations. For welfare considerations, we wanted the prototyped system to balance both power-efficiency and welfare. As such, we opted for an NVIDIA Jetson Nano - a middle ground between a power-inefficient computer and an inexpensive, yet less powerful FPGA. This also contributes to our consideration of environmental factors, as we wanted to minimize our carbon footprint by the technology stack our system would use. This meant moving away from computationally-heavy machine learning models, and opting for embedded processors and microcontrollers over heavier, heat-generating computers. The social considerations were built into how we implemented the system. For a good product experience, users want to see instantaneous feedback and interactivity. This is why we stressed the importance of keeping the end-to-end processing per frame to be under 100ms. If we had additional delay or relaxed this constraint too much, the usability of the product would be far less. Another big concern we had was with safety, especially that regarding the camera and projector mount. Since both objects are fairly heavy and are fixed well-above the table, it would be dangerous to construct our own camera/projector mount. If the structure is unstable, the debris, projector, and camera could injure players using the system. As such, we opted to purchase an existing, sturdy metal mount that can support far more weight than the projector and camera to ensure safety of our system.

VII. Test, Verification and Validation

Our project will primarily employ three methods that measure three metrics to evaluate the design implementation: 1) latency, 2) trajectory prediction accuracy, 3) object detection accuracy.

For testing the latency, we plan on timing the code execution for each frame programmatically. As soon as a frame comes in from the camera via OpenCV, we will include with the frame the marked input time as additional metadata. When the frame finishes processing, we will again take the time when the predictions are generated and find the difference (in milliseconds). Since the software system is entirely contained within the laptop, we do not need to account for transmission latency via Wi-Fi. Our use case requirement regarding latency was that end-to-end processing would be within 100ms. By ensuring that each individual frame takes less than 100ms to be processed through the entire software pipeline, we ensure that the user perceives changes to the projected predictions within 100ms which has the appearance of the predictions being instantaneous.

For testing the trajectory prediction accuracy, we plan to manually verify this. The testing plan is as follows: an experienced pool player will take 10 shots straight-on, and we will use the mounted camera to record the shots. Then, we will manually look through the footage and trace out the actual trajectory of each ball struck and compare it to the predicted trajectory. The angle between these two trajectories will be the deviation. Angle deviation from all 10 shots will be averaged, and this will be our final value to compare against the use-case specification of <= 2 degrees error. In order to account for error in the player's technique and accuracy, we take the average deviation over multiple iterations to smoothen out this error from the measurement. Thus, the test captures the actual trajectory deviation with fair accuracy and will validate the use-case requirement of having less than 2 degrees of angle deviation error.

For object detection accuracy, the test we have planned will measure the offset between the projected object and the real object on the pool table. Since the entire image will be scaled the same, we will use projections of the pool balls in order to test the object detection accuracy. The test will be conducted as follows: we will set up the pool table in some arbitrary configuration, then take in a frame and run our computer vision system on it. Then, we will project the modified image with objects detected over the actual pool table. We will measure the

distance between the real pool balls and projected pool balls (relative to each ball's center) by taking a picture and measuring the difference manually. This will ensure that our object detection model is accurate and correctly captures the state of the pool table.

VIII. Project Management

A. Schedule

Our schedule for the project is depicted in Fig. 4 on page 11 of this report. The blocks in blue represent the sections that are being done by Andrew, green represents the sections being done by Tjun Jet, and the red block represents the sections being done by Debrina. The yellow blocks indicate the blocks that will be done by everyone. Our schedule includes weekly milestones and allows us to keep track of what has been done, what is currently in progress, and what has not been completed.

B. Team Member Responsibilities

Debrina's primary responsibility is the Computer Vision subsystem. She is responsible for implementing object detection and segmentation algorithms. The objects that will have to be clearly segmented are the array of pool balls on the table, distinguishing between solid and stripes, cue pockets, cue stick, and the four walls surrounding the table. She will have to ensure the detections are accurate enough to meet our use case requirements. She is also in charge of the camera that we buy, and making sure the specifications are enough to perform the computer vision algorithms.

Tjun Jet's role is to work on the physics engine. The physics engine takes in the outputs from Debrina's computer vision model and calculates the trajectory that the cue ball will follow if it's being hit in a certain manner. He will focus on two main detection models - wall detection and ball collision mechanics. This means that he must take into account the reflections of the wall, and the mechanics when the cue ball collides with the ball. He then creates a single line as an output to the projector subsystem.

Andrew will play a major role in the projector subsystem and the IMU subsystem. He will take the output from Tjun Jet's physics model as input to his projector subsystem. Then, he will put it into a relevant frame to project the line onto the table. The crucial role he must play is ensuring that this line is accurate and well calibrated to the table, making any adjustments necessary if needed. He will also be responsible for taking the IMU data and streaming it onto our web application, to provide recommendations for users in terms of their shot force, angle, and spin.

C. Bill of Materials and Budget

Our bill of materials and budget can be found in Table I on page 10 of this report.

D. Risk Mitigation Plans

A difficult task in our project is ensuring that our model meets our use-case requirements. One risk we might encounter is that the latency of our detection becomes very high, making the user interface not as good as we want it to be, as the output trajectories will be too slow. Furthermore, we might also face a problem where a small error in angle and trajectory could lead to a significant error in execution of the actual pool shot. For object detection inaccuracies and/or large latencies, we aim to rely more heavily on April Tags to mitigate these issues. For instance, if the pool table cue detection is inaccurate or it takes too long to identify it, we can attach April Tags to the cue stick. Then, we can automatically detect the tip of the cue stick and save a lot of computation time by replacing our otherwise complex CV algorithms for cue stick detection. Additionally, we can mitigate latency issues by increasing the power of our hardware. If the processing time end-to-end significantly exceeds 100ms, we can opt to use a computer instead of the Jetson Nano to speed up the prediction process. Then, if the latency between wireless devices is too high, we can mitigate this by switching to wired connections instead of wireless.

IX. Related Work

A project that is similar to ours was done by Team C7: 8-Ball Lifeguard during the Spring 2023 semester for CMU's ECE Capstone course [3]. This project's use case was also to help their users learn to play pool. The project gave players a single suggestion on which ball to hit, and from which angle, based on the state of the balls on the pool table. This recommendation would be projected onto the pool table, and the users were expected to follow the suggested hit. While this project has similar functionality to ours, we prioritize the interactive aspect of our proposed project as we believe it is more effective in helping users improve in eight-ball pool.

X. Summary

Our project is an assistive tool that helps users learn eight-ball pool more effectively, without the need of others' instruction. Our system guides users who are playing a game of pool by showing users a projection of the trajectories of the target ball they are aiming at. This trajectory is based on the orientation with which the user is holding the cue stick. The trajectories, which are projected onto the table using a projector that is mounted overhead, will allow the user to see whether their current aim would allow them to successfully pocket a target ball. Our project emphasizes a responsive user interface, so as the user moves the cue stick around, the user will be able to see the trajectories readjust accordingly. An IMU attachment on the pool cue would provide further feedback to the user on the velocity and spin of their hit. Our system uses computer vision models to detect the state of the pool table and processes these frames using a physics model that we have developed ourselves. Some challenges that may emerge in the implementation of our design is ensuring the accuracy of our predicted trajectories and reaching a target latency of 100ms. However, we have created a plan to tackle these risks if needed by using April Tags to provide more accurate object detection and using a computer to run our models with a lower latency.

GLOSSARY OF ACRONYMS.

CV – Computer Vision
FPGA – Field Programmable Gate Array
HTML – Hypertext Markup Language
IDE – Integrated Development Environment
IMU – Inertial Measurement Unit
USB -- Universal Serial Bus

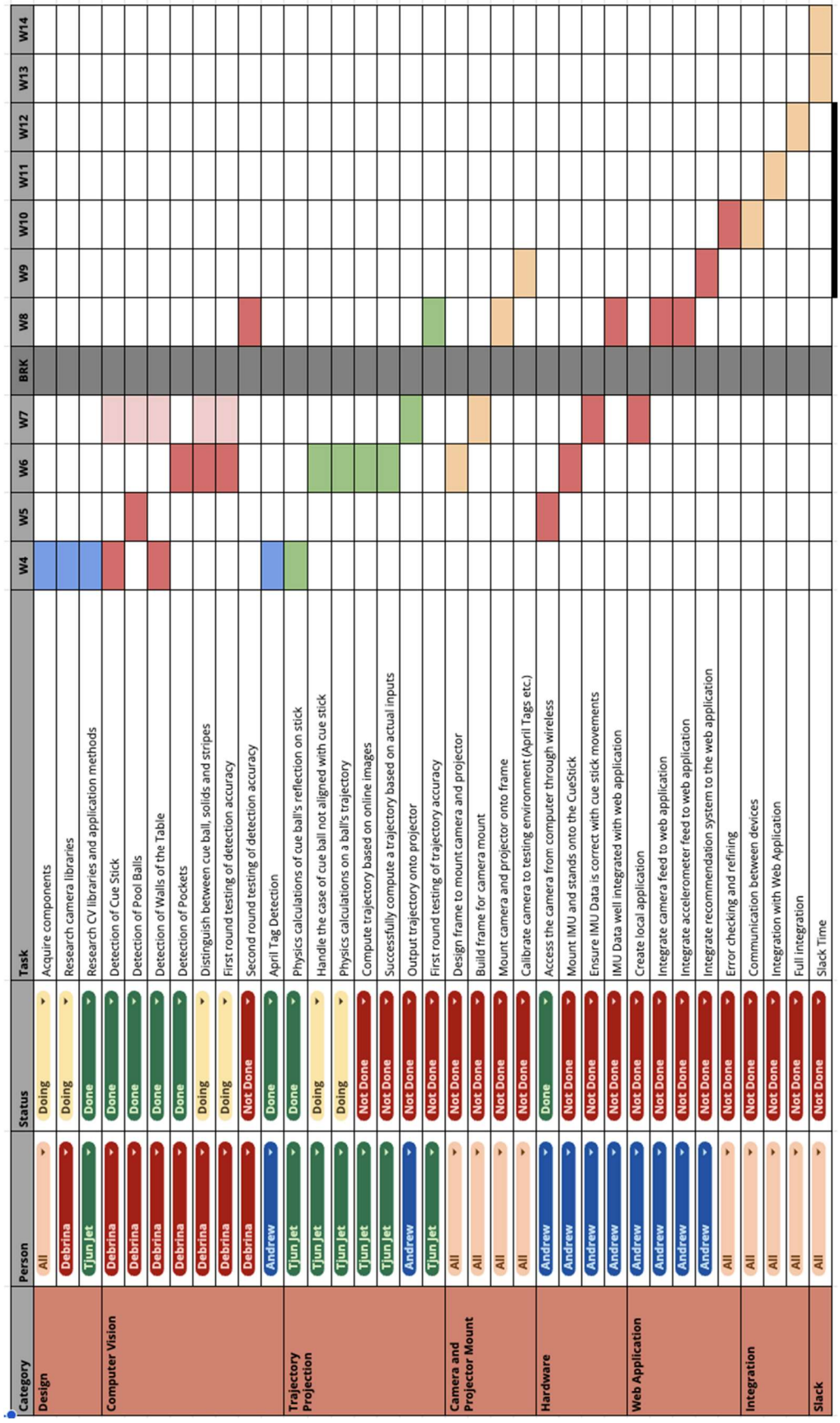
REFERENCES

[1] R. B. Miller, "Response time in man-computer conversational transactions," in Proc. AFIPS Fall Joint Computer Conference, vol. 33, pp. 267-277, 1968.
[2] AZBilliards Forums, "Fractional Aiming and Required Accuracy," Available: https://forums.azbilliards.com/threads/fractional-aiming-and-required-accuracy.522183/.
[3] Agarwal, Rager, Ray "Team C7: 8-Ball Lifeguard" Carnegie Mellon University. Available: https://course.ece.cmu.edu/~ece500/projects/s23-teamc7/.

TABLE II. BILL OF MATERIALS

Item	Part Name	Manufacturer	Quantity	Cost @	Total
Pool Table	RayChee Portable Mini Billiard Table	RayChee	1	$129.99	$129.99
Rack	Muscle Rack 5-Shelf Steel Freestanding Shelving Unit, Black	Muscle Rack	1	$109.00	$109.00
WiFi-enabled Microcontroller	ESP-WROOM-32 ESP32 ESP-32S Development Board	AITRIP	1	$15.99	$15.99
Inertial Measurement Unit	Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055	Bosch	1	$34.95	$34.95
Embedded Processor	NVIDIA Jetson Nano	NVIDIA	1	$0	$0
Laptop	Macbook Air	Apple	1	$0	$0
Camera	Logitech C922 Pro Stream	Logitech	1	$0	$0
Battery	9V Batteries	Amazon Basics	1	$0	$0
Projector	VOPLLS 1080P Full HD Mini Projector	VOPLLS	1	$49.99	$49.99
LED Light Strips	Tenmiro 65.6ft Led Strip Lights	Tenmiro	1	$9.99	$9.99
				Grand Total	**$349.91**

FIG. 4. PROJECT SCHEDULE



Category	Person	Status	Task	W4	W5	W6	W7	BRK	W8	W9	W10	W11	W12	W13	W14
Design	All	Doing	Acquire components												
	Debrina	Doing	Research camera libraries												
	Tjun Jet	Done	Research CV libraries and application methods												
Computer Vision	Debrina	Done	Detection of Cue Stick												
	Debrina	Done	Detection of Pool Balls												
	Debrina	Done	Detection of Walls of the Table												
	Debrina	Done	Detection of Pockets												
	Debrina	Doing	Distinguish between cue ball, solids and stripes												
	Debrina	Doing	First round testing of detection accuracy												
	Debrina	Not Done	Second round testing of detection accuracy												
	Andrew	Done	April Tag Detection												
Trajectory Projection	Tjun Jet	Done	Physics calculations of cue ball's reflection on stick												
	Tjun Jet	Doing	Handle the case of cue ball not aligned with cue stick												
	Tjun Jet	Doing	Physics calculations on a ball's trajectory												
	Tjun Jet	Not Done	Compute trajectory based on online images												
	Tjun Jet	Not Done	Successfully compute a trajectory based on actual inputs												
	Andrew	Not Done	Output trajectory onto projector												
	Tjun Jet	Not Done	First round testing of trajectory accuracy												
Camera and Projector Mount	All	Not Done	Design frame to mount camera and projector												
	All	Not Done	Build frame for camera mount												
	All	Not Done	Mount camera and projector onto frame												
	All	Not Done	Calibrate camera to testing environment (April Tags etc.)												
Hardware	Andrew	Done	Access the camera from computer through wireless												
	Andrew	Not Done	Mount IMU and stands onto the CueStick												
	Andrew	Not Done	Ensure IMU Data is correct with cue stick movements												
	Andrew	Not Done	IMU Data well integrated with web application												
Web Application	Andrew	Not Done	Create local application												
	Andrew	Not Done	Integrate camera feed to web application												
	Andrew	Not Done	Integrate accelerometer feed to web application												
	Andrew	Not Done	Integrate recommendation system to the web application												
	All	Not Done	Error checking and refining												
Integration	All	Not Done	Communication between devices												
	All	Not Done	Integration with Web Application												
	All	Not Done	Full integration												
Slack	All	Not Done	Slack Time												