Slowpitch Softball Pitch Detector

Design Document

sdmay25-11
Nick Fila
Dr. Jones
Andrew Vick (Machine Learning Integration)
Casey Gehling (Client Interaction)
Sullivan Fair (Individual Component Development)
Ethan Gruening (Team Organization)
Josh Hyde (Research)
Cameron Mesman (Testing)
Team Email
https://sdmay25-11.sd.ece.iastate.edu/

Revised: 12/3/24

Executive Summary

Our senior design project focuses on developing a system to assist umpires in detecting illegal pitches in slow-pitch softball games. The problem arises from the challenge of accurately determining whether a pitch falls outside the legal bounds of 10 to 12 feet in height, leading to inconsistent calls and disputes. Our solution aims to eliminate this ambiguity, allowing umpires to focus on other aspects of the game while ensuring fairness and accuracy.

To address this issue, we are designing a mobile application that leverages computer vision and machine learning to analyze softball pitches in real-time. This approach removes the need for leagues to invest in expensive or specialized hardware, making our solution cost-effective and accessible to a wide range of users. The app uses OpenCV and YOLO (You Only Look Once) for object detection and tracking, enabling it to identify the softball and monitor its trajectory throughout the pitch.

Key design requirements include accuracy in determining pitch legality, minimal interference with the flow of the game, and affordability for users. Feedback from current umpires has validated these priorities, and our design aligns with their expectations.

Thus far, we have implemented C++ code for detecting and tracking the softball using YOLO and OpenCV. Additionally, we have developed calibration functionality that allows the user to define key reference points—such as the pitcher's mound and home plate—and input the camera's height. This calibration maps the physical space into a frame of reference, enabling precise height calculations for the pitch and determining whether it is legal.

Our design meets the defined requirements by ensuring the app runs smoothly on readily available smartphones, avoiding the need for additional hardware. Accuracy is achieved through thorough testing and adjustments in calibration and tracking. The next steps include integrating the calibration and tracking code into the Flutter app, creating a seamless user interface, and conducting comprehensive real-world testing to further refine performance.

By enabling umpires to confidently identify illegal pitches, our product aims to improve the quality and fairness of games while maintaining accessibility and usability for all levels of softball leagues.

Learning Summary

Summary of Requirements

- Object detection system to locate a softball at its maximum height during a pitch
- Detect an illegal pitch when a softball's maximum height is higher than the specific maximum height or below the specific minimum height.
- Create an audible sound indicating an illegal pitch
- The device cannot be physically obstructive to the game.
- The device must accommodate different fields, lighting, and balls.
- The device cannot be visually distracting to the game.
- The device must have a guided and simple setup.
- The device must have user-adjustable settings for the maximum and minimum height for pitches.

New Skills/Knowledge acquired that was not taught in courses

- Computer Vision
- Machine learning applications
- Flutter framework
- iOS development

Table of Contents

1. Introduction

1.1. Problem Statement

In the game of softball, one crucial rule states that a pitch is only legal if the ball reaches its peak height within a minimum and maximum height bound during its flight to the batter. Currently, umpires must estimate this height by eye for every single pitch: a challenging task that relies heavily on personal judgment and can vary from one umpire to another. This reliance on subjective estimation can lead to inconsistent calls, potentially altering the outcome of games and affecting the fairness and enjoyment of the sport. Players might strike out on pitches that should have been deemed illegal, and fans may witness games swayed by questionable decisions. These issues highlight a broader problem: the lack of accessible technology to assist in making precise, real-time measurements during games. To address this, we are developing an affordable and user-friendly device that tracks each pitch using a single camera, accurately measures the ball's height using machine learning, and alerts officials when a pitch falls outside the legal bounds. By eliminating the guesswork from pitch height estimation, we aim to enhance the accuracy of calls, uphold the integrity of the game, and improve the experience for players, umpires, and fans alike.

1.2. Intended Users

Our product is designed to benefit several key groups within the softball community, including umpires, players, fans, and coaches.

1. **Umpires**
    a. **Description**: Umpires are the officials responsible for enforcing the rules of softball during games. They must make quick, accurate decisions under the pressure of real-time play, often without technological assistance.
    b. **Needs**: Umpires need a reliable method to accurately determine if a pitch meets the legal height requirements, reducing the reliance on subjective judgment and minimizing errors that could impact the game's outcome.
    c. **Benefits**: Our device provides umpires with precise, real-time measurements of each pitch's height. This technology aids them in making consistent and accurate calls, reducing stress and enhancing their confidence in officiating. By supporting umpires with accurate data, we help maintain the game's fairness and integrity, directly addressing the issues outlined in our problem statement.
2. **Players**
    a. **Description**: All players in a softball game are directly affected by the calls made by umpires and strive for a fair and competitive environment to showcase their skills.
    b. **Needs**: Players need assurance that the rules are being applied consistently so that their performance is judged fairly. Batters, in particular, need protection from illegal pitches that could unfairly result in strikeouts.
    c. **Benefits**: With our device ensuring accurate detection of illegal pitches, players can trust that the game is being officiated fairly. Batters are less likely to be unfairly penalized, and pitchers receive clear feedback on the legality of their pitches. This fosters a fair playing field and allows players to focus on their performance, enhancing the game's overall quality in line with our problem-solving goals.

3. **Fans**
    a. **Description**: Fans are the spectators who enjoy watching softball games, whether in person at the stadium or through broadcasts. They are passionate about the sport and value exciting, fair competition.
    b. **Needs**: Fans desire an enjoyable viewing experience where player skill rather than officiating errors determine the game's outcome. They appreciate transparency and fairness in how the game is played and called.
    c. **Benefits**: By improving the accuracy of pitch legality calls, our device enhances the fairness and excitement of the game, leading to a more satisfying experience for fans. They can enjoy the sport knowing that technology is helping to uphold its integrity, which aligns with our aim to improve the overall enjoyment of softball.
4. **Coaches and Teams**
    a. **Description**: Coaches and team staff are responsible for training players and developing game strategies. They work closely with players to improve skills and ensure compliance with the rules.
    b. **Needs**: Coaches need effective tools to train pitchers on delivering legal pitches and to adjust strategies based on accurate information about gameplay.
    c. **Benefits**: Our device can be used during practices to provide immediate feedback to pitchers on the height of their pitches, aiding in skill development and rule compliance. During games, accurate officiating supported by our product allows coaches to focus on strategy without worrying about inconsistent calls. This directly enhances player performance and upholds fair competition, as highlighted in our problem statement.

By serving these users, our product addresses the critical issue of subjective pitch height estimation in softball. It promotes fair play, enhances the accuracy of officiating, and improves the overall experience for everyone involved in the sport. Our solution connects directly to our overarching goal of removing guesswork from the game, ensuring that softball is enjoyable, fair, and competitive for all participants

2. Requirements, Constraints, And Standards

2.1. Requirements & Constraints

Functional Requirements

- Object detection system to locate a softball at its maximum height during a pitch.
- Detect an illegal pitch when a softball's maximum height is higher than the specific maximum height or below the specific minimum height. (constraint)
- Create an audible sound indicating an illegal pitch

Physical Requirements

- The device cannot be physically obstructive to the game.
- The device must be portable to set up on a softball field.

Environmental Requirements

- The device must accommodate different fields, lighting, and balls.
- The device cannot be visually or audibly distracting to the game.

UI Requirements

- The device must have a guided and simple setup.
- The device must have user-adjustable settings for the maximum and minimum height for pitches. (constraint)

Constraints

- Maximum height (in feet) the softball can reach on a serve.
- Minimum height (in feet) the softball must reach on a serve.
- The device cannot be physically obstructive to the game.

2.2. Engineering Standards

The Importance of Engineering Standards (Q1)

Engineering standards are important because people interact with engineering products every day. If there were no engineering standards, that would not only compromise the desired quality of engineering products but also affect things like the safety of everyday people. For example, people interact with civil engineering like bridges and buildings, and non-physical structures like apps and programs. A lack of requirements for these products could result in catastrophic failures of civil structures or a breach of confidential personal data from an insecure application. Engineering standards help ensure that these types of issues are avoided so people can continue to use these products to improve their lives and the lives of others

Published Engineering Standards (Q2)

- **IEEE 1857.9-2022** provides standards for video encoding/decoding and analyzing methods. These standards apply to all forms of video manipulation, spanning from areas such as network video transmission over services such as UDP to computer vision topics such as object detection.
- **IEEE P3110** provides the standards for the necessary API requirements in a computer vision implementation. These API abstractions interface with various machine learning algorithms and are used to develop computer vision solutions such as OpenCV.
- **IEEE 1008-1987** provides guidelines for proper unit testing with a codebase. Specifically, guidelines exist for proper categories of testing (unit, integration, etc.) and code coverage reporting, among other testing-related requirements.

Engineering Standards Relevancy (Q3)

After reviewing the three standards, each has varying relevance to our project. Since our primary focus is on computer vision, IEEE P3110 is highly applicable. This standard provides guidelines for API requirements in computer vision and will help us ensure our project adheres to best practices when working with machine learning algorithms like OpenCV. It offers a strong foundation for structuring our computer vision solution. IEEE 1008-1987 also holds some relevance, as it provides useful guidelines for software testing. While our project's smaller scale, incorporating structured unit testing can still improve code reliability. On the other hand, IEEE 1857.9-2022 is less applicable since it focuses more on video encoding and compression, which isn't central to our object detection work.

Other Applicable Standards (Q4)

Some of the standards we found that differed from those provided that might be at least somewhat applicable to our project are below. IEEE Standard Digital Interface for Programmable Instrumentation. This standard applies because we use equipment and information that takes in different measurements. We will use different types of instrumentation that may be programmable to detect the softball. IEEE Standard for Application Programming Interfaces (APIs) for Deep Learning (DL) Inference Engines. This engineering standard is applicable because we are programming a solution to consistently and accurately find the height of a softball, including deep learning and AI-based detection algorithms. IEEE Standard Letter Symbols for Units of Measurement (SI Customary Inch-Pound Units, and Certain Other Units). This standard is applicable because it ensures that we incorporate the appropriate symbols for our measurement and that it is understood clearly using accurate measurements and symbols.

Modifying Our Project to Incorporate Engineering Standards (Q5)

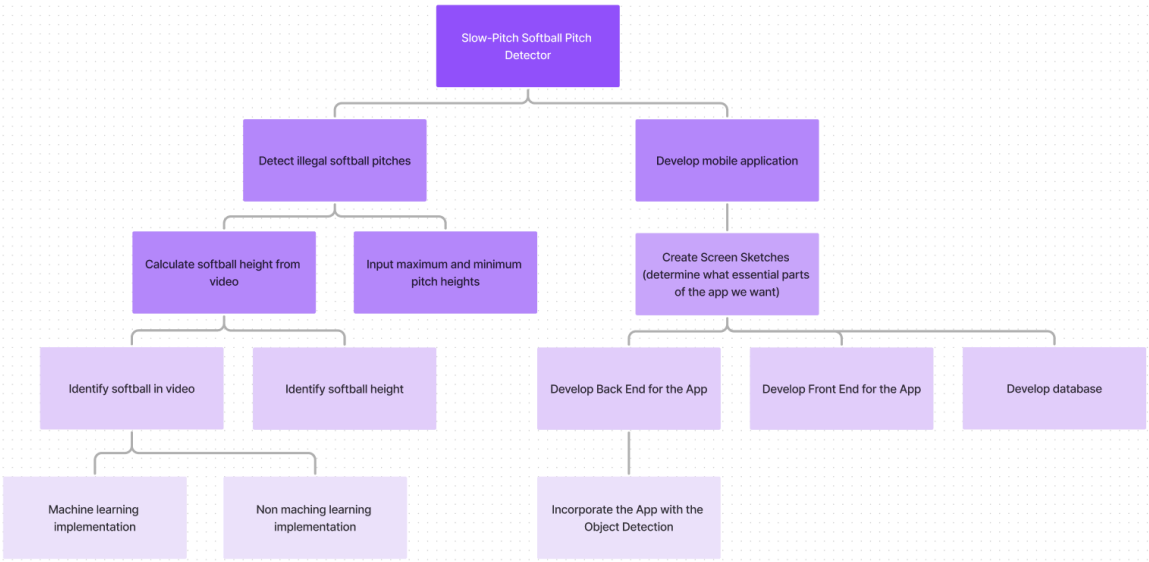
Since our design is not finalized as we continue to test and prototype different designs, we don't need to make any modifications now. However, several of these standards must be considered as we proceed with our design. Firstly, as we develop the software component of the project, we will need to keep in mind IEEE 1008-1978 regarding unit testing. We need to ensure that we are writing the software in a way that can be easily tested. We will also need to consider IEEE P3110 which covers standards with computer vision. Since many of us have never built an official project using computer vision, it is not natural to consider these standards. Moving forward, standards like this one must be considered during design

3 Project Plan

3.1 Project Management/Tracking Procedures

We decided to adopt an AGILE methodology to manage our project. There are complex components we need to manage with our app. These components include our height detection script utilizing a hybrid machine and non-machine learning approach and a mobile app utilizing Flutter and Dart FFI. To make development more manageable, we can divide the work into AGILE sprints with smaller goals, so we are not trying to tackle the entire project at once. This will improve our understanding of individual project components and help make a full implementation easier. To keep track of our progress, we are utilizing Git issues coupled with personal branches for individual development. Having an issues board will help us manage what tasks must be done. Using personal branches will help isolate development so we can develop individual components more efficiently without accidentally breaking the project's main branch or having more than one person alter the same file in different ways.

3.2 Task Decomposition



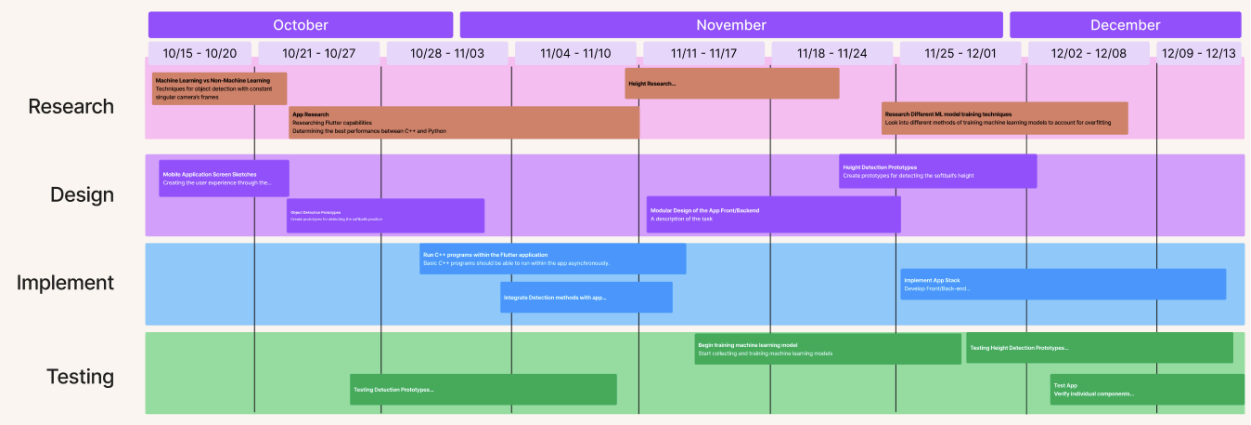
These tasks are intentionally left broad since we have not made final decisions on the entire design. Based on the agile methodology, many of these tasks will be adjusted as development continues and we receive feedback from the client. Any testing and prototyping we do could also affect our final implementation of these tasks. With this task decomposition, we split the project into two major tasks: detecting illegal pitches and developing the mobile application. These two tasks pretty much sum up our project but are still broken down further in the image above.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 3.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

3.4 Project Timeline/Schedule



Above is our proposed AGILE project timeline, featuring a clearly outlined sequential list of milestones. Our project has been separated into four parallel-running distinct categories: the research category, in which project specific applications and approaches are thoroughly assessed and evaluated, the design category, where compiled research is initially primed for development, the implement category, where designs created are prototyped and implemented into real applications, and the testing category where implementations are tested for their correctness and efficiency. Following an AGILE methodology, we have separated our project timeline into roughly one-week sprints where certain tasks are scheduled for completion. Each bar on the above Gantt chart represents a deliverable within an AGILE timeframe set to expire at the bar's end. Holistically, the above Gantt chart can be summarized in sprints as follows:

Timeframe/Deliverables

10/15 - 10/20 (Sprint 1)

- Machine learning vs. non-machine learning research
- Mobile application screen sketch design

10/21- 10/27 (Sprint 2)

- Flutter app research
- Object detection prototype

10/28 - 11/03 (Sprint 3)

- Object detection prototype testing
- Flutter and C++ integration research and development

11/04 -11/10 (Sprint 4)

- Implementation of object detection within mobile application

11/11 -11/17 (Sprint 5)

- Height calculation research
- Modular design of full-stack application
- ML model testing

11/18 -11/24 (Sprint 6)

- Height detection prototyping

11/25 - 12/01 (Sprint 7)

- ML model training techniques research (model refinement)
- Full stack application implementation

12/02 - 12/08 (Sprint 8)

- Height detection testing

12/09 - 12/13 (Sprint 9)

- Mobile app integration testing

3.5 Risks and Risk Management/Mitigation

Sprint 1:

- Risks:
  - ML may be overkill for object detection and too resource-intensive. (prob: 0.5)
  - Non-ML methods might lack the needed accuracy. (Prob: 0.6)
- Mitigation:
  - Explore using a combination of ML and non-ML strategies. ML can be used to check our non-ML tracking to correct for deviation.

Sprint 2:

- Risks:
  - Flutter's limitations in handling real-time processing or specific hardware task. (Prob: 0.3)
  - Object Detection prototype may not achieve the required performance or accuracy. Also may not integrate well with Flutter. (Prob: 0.9)
- Mitigation:
  - Rework prototype to use (C++/OpenCV) instead of Python.

Sprint 3:

- Risks:
  - Compatibility issues with FLutter and C++ integrations, particularly in efficient data passing. (Prob: 0.5)

        - Performance drop when integrating object detection into the mobile interface. (prob: 0.4)
- Mitigation:
    - Use Native Flutter plugins if integration lags. Test different data-handling techniques (i.e. JSON or shared libraries) to see what works best.

Sprint 4:

- Risks:
    - Computing power of mobile devices may limit detection effectiveness. (Prob: 0.6)
    - App design may require significant performance tuning to be responsive. (Prob: 0.5)
- Mitigation:
    - If mobile limitations are severe, explore using lightweight detection models like TensorFlow Lite.
    - Set up performance tests in this sprint to identify bottlenecks.

Sprint 5:

- Risks:
    - Height calculations may require a more complex algorithm than anticipated. (Prob: 0.5)
    - Modular design complexity might extend timelines. (Prob: 0.4)
- Mitigation:
    - Prioritize simple algorithms and add complexity only as needed.
    - Break down modular design into manageable components and integrate one at a time.

Sprint 6:

- Risks:
    - Height detection requires more data points or higher resolution. (Prob: 0.5)
    - Prototype might need hardware not accessible or feasible on mobile devices (Prob: 0.3)
- Mitigation:
    - Consider using simpler relative height detection if absolute values are challenging.
    - Explore OpenCV's scaling and resolution techniques.

Sprint 7:

- Risks:
    - Model training techniques could require extensive datasets not readily available. (Prob: 0.6)
    - Full-stack implementation might increase overhead and limit application speed. (Prob: 0.4)
- Mitigation:
    - Consider using pre-trained models to limit the computational needs of training a new model.

Sprint 8:

- Risks:
    - Detection algorithms may not generalize well across different test conditions. (Prob: 0.5)
- Mitigation:
    - Run multiple rounds of testing in varied lighting/angles and adjust thresholds. Update algorithms or training data based on observed issues.

Sprint 9:

- Risks:
    - Final app integration may encounter unexpected platform constraints, e.g., compatibility issues on different devices (Prob: 0.5)
- Mitigation:
    - Use platform-specific checks to handle potential discrepancies.
    - Testing should include various devices, screen sizes, and operating systems.

3.6 Personnel Effort Requirements

Team Member	Task Descriptions	Hours Worked
Sullivan Fair	Prototype object detection scripts, Screen sketches, MOSSE object detection script, C++/Flutter integration research, C++/Flutter integration prototype, Initial Flutter screens, Created OpenCV/Tracking framework, C++ tracking code reduction	57
Casey Gehling	Flutter screen development, C++/Flutter integration research, codebase hygiene maintenance, field research, video caching implementation, client interaction/team interaction	59
Ethan Gruening	Prototype object detection scripts, train a custom YOLO model, research existing products, collect pitch videos, prototype color calibration scripts, prototype camera undistortion scripts, create a Flutter camera setup to determine known heights, set up automatic compilations for iOS and Android.	70
Josh Hyde		
Cameron Mesman	Flutter screen mock-ups, integrating C++ code into flutter, integrating illegal pitch audio into flutter code	50
Andrew Vick	Prototype object detection code, translate python scripts into C++, prototype detection and tracking code	

	running on iPhone, Integrate tracking code and object detection	

3.7 Other Resource Requirements

Developing a mobile application, our team already owns cell phones to test our Flutter applications on iOS and Android operating systems. The primary resources our team has and will continue utilizing are Iowa State's recreational softball fields. Field availability is crucial for the research and testing for proper camera setup, object detection, and in-game functionality. Field reservations will be made as needed for future testing.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	**Description**	**Examples**
Public health, safety, and welfare	Our project is a bystanding officiary to an existing softball game, an environment with a high risk for player injury. Our design must be non-intrusive and not be in the way of gameplay. There should be little to no contact between the game activity and our physical device to ensure public safety.	Considering public safety, our setup is a mobile device placed on a mount along the fence. This ensures minimal physical interaction between the players and the device, limiting additional risk for the players.
Global, cultural, and social	Slow-pitch softball is a variation of a typical softball game for more novice players. Local leagues and amateur slow-pitch players are our target communities and it is our mission to ensure this device is catered to their financial and accessible needs.	As engineers, our social and ethical responsibility is to create a design to accommodate local leagues. Our design as a mobile application opens our user base to all people with smartphones. This addresses the low-cost and portable solution that local leagues would require.
Environmental	Since our project is very niche, there is little to no environmental impact when using our product. However, it is important to recognize the environmental impact of materials and power consumption of the operating devices.	Our product can be downloaded onto a user's smartphone to limit additional electrical components. Additionally, a power supply for the smartphone will need to be provided and may require a battery charger for prolonged extended use.
Economic	Catering to local slow-pitch leagues, we must consider how our product will impact the league's funds for both umpire training and operation.	Our product will be affordable for all users within a slow-pitch league since it requires an existing smartphone. A simple setup

		ensures that there will be little time training umpires to use the application or set it up before a game.

4.1.2 Prior Work/Solutions

Include relevant background/literature review for the project (cite at least 3 references for literature review in IEEE Format. See link: https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf)

Beginning our discussions on our project design, we researched existing products and solutions for cameras that can track balls within a sports setting. Analyzing the existing products, their user reviews, and their targetted impact helped our team decide what approach would best suit our project.

The first product we researched was the Specialized Imaging Tracker 2, a high-speed fast projectile measurement camera. This camera has three-axis rotation for camera positioning and is programmable for customization. Many users liked the full remote operation, multiple operating modes, tight accuracy error margins, and no need for calibration. However, this is a research-grade camera with a high cost and is very large. Considering our use case, local slow-pitch leagues would appreciate a camera with small error bounds and little to no calibration. Additionally, it's essential for our users to have an affordable and portable option for recreational use. The Specialized Imaging Tracker 2 may be too advanced for the operating scenario and out of the price range for local leagues.

A similar product used in professional sports games is the Veo 3 Sports Cam. This is a 360-degree camera that not only tracks sports balls but also tracks players, displays heat maps, match stats, and tags game actions. This device is known for its object detection accuracy and long battery life. Users can also live stream their sports games through this camera. These features would benefit local softball leagues, though they are not needed for operating as a slow-pitch officiary. The downside to this product is it costs $2,400 and requires a subscription to operate. Used in a professional sense, the Veo 3 Sports Cam is a successful product. Its features may not be needed for the primary operations of a slow-pitch officiary tool; however, the high standard for preserving the integrity of the game with an accurate system gives users the confidence to use the product in game settings.

The Pocket Radar Smart Coach is another product specializing in ball detection in a softball setting. This product is a small device, about the size of a smartphone, and records a clip of a softball pitch and its speed. The device is compatible with a mobile app that saves the video

and speed of pitches. With a strong user interface, there are many positive user reviews. Users typically buy this product for the connection to their mobile device, the ability to share and review their pitches, and a cost-effective solution with a portable design. The Pocket Radar targets users who recreationally play and practice softball and baseball. Their isolation of extra sensors and cameras into a handheld device makes it accessible. A drawback to creating a small device is there is low battery life that limits users to a small usage time. The lower-quality camera has a more significant error bound than professional cameras, with a 1 mph error. Although this device has a small battery life and a more significant error bound, it is more tailored for local and recreational users, receiving positive feedback for their specific use cases.

4.1.3 Technical Complexity

Our slow-pitch softball application has several components that integrate together mathematical, engineering, and social challenges. Our engineering challenges can be split into two main modules: the frontend user experience and the height detection. The project's technical complexity would be a large project to design, prototype, and test. Considering this needs to be developed to be affordable and portable for local leagues, this increases the complexity as more readily available technologies should be used for a wider user base with lower costs.

The height detection component of the project will require using a camera feed and multiple object detection techniques and existing libraries in tandem. Additionally, object detection alone will not be enough to determine height; a technique must be found to coordinate the ball's location to a specific height through geometric calculations or calibrated values. With the limited existing products, our team must find our own solution to determining height using engineering principles and practices. Once accomplished, there can be further development to increase the fps or computation time for frame analysis. As an open-ended component of the project, the technical design, prototyping, and testing is a true example of an engineering problem within the real world.

The height detection backend component is the application's "computational brain." Gathering calibration information, collecting camera frames, and setting maximum and minimum heights will need to be incorporated into the system's frontend display. A user would interact with the program in several ways to start, end, and view the officiary logs. It is essential to develop an accessible and guiding frontend for umpires and players to experience a seamless setup and collection process. The compatibility of the frontend and the backend is also very important. This module is a technically advanced project, not only developing a user interface but also stitching it with the backend's operations and reporting its output.

In total, the technical components of this project provide complex engineering problems that match the industry standards as a real-world software project.

4.2 Design Exploration

4.2.1 Design Decisions

A primary design decision was what system we would use for the camera collection and user interface. This decision will define what our users will need to purchase, calibrate, and gain familiarity with when running our application. From a technical standpoint, using multiple cameras would gain more data points to pinpoint the exact height of the ball. However, developing this product specifically for the ease and affordability of local slow-pitch leagues, we decided to implement this application on a singular mobile device. This design decision increases the project's technical complexity but better suits our user base. The application was chosen to be developed using Google's app development platform, Flutter, which can be compiled to run on both Android and iOS. Allowing anyone with a mobile device to install our application invites many users to interact with the widget-based accessible program. This was our most critical design decision, led by our social and ethical responsibility and our user needs.

Another design decision was the technologies to use in the object detection of the softball within the camera frame. This decision was based on the compatibility with Flutter, the speed of the library's methods, and the accuracy. After careful consideration and research, we chose C++ as our programming language for object detection. It is a very fast language with several libraries, such as OpenCV, to analyze an image. C++ can also be integrated into a Flutter application and be configured to run on iOS and Android to sustain our cross-platform development. We can use OpenCV and its KCF algorithm methods to take in an image frame, find a colored object, and track its movement over time. Integrating a YOLO machine learning model to track sports balls also provides a stronger and more accurate collection method to validate OpenCV's tracking. Additionally, the decision to use OpenCV and YOLO within a C++ project was heavily influenced by the existence of these libraries in Python. This allows our team to prototype object and height detection techniques at the beginning of our project's design stages. Converting our chosen design to C++ with these libraries can elevate our project and encourage multiple prototype designs.

The final major design decision was the height calculation technique used to take in the position of a softball within a camera frame and output a specific height. After researching, the vertical pixel distance is constant and can be converted to feet when calibrated with known heights and a horizon line (the line from the home plate to the pitcher's mound). This was found by placing a camera equidistant from the home plate and the pitcher's mound, selecting where both are located within the frame and where a known height exists along the horizon line. The image below shows the equal segments representing 2ft.

We now have a technique to analyze the (x,y) of the softball and determine if it lies within the maximum and minimum height bounds set by the user.

4.2.2 Ideation

The design decision we struggled with the most was finding and deciding on the height detection technique. As mentioned in section 4.2.1, we finalized the decision for finding the corresponding ground line below the pitch from home plate to pitcher's mound and identifying a known height to get the vertical pixel-to-feet ratio and turn the detected softball's (x,y) coordinates into a height value. There were other prototyped and researched height calculation techniques that we considered for the final design

- Multi-Camera Detection
    - Set up three cameras in different locations throughout the field.
    - Run the object detection on all the cameras and determine the ball's distance from each camera with the camera's detected radius.
    - You can use each camera's height to triangulate the exact location of the ball, including height.
    - This violates our design choice for a single camera.
- Machine Learning
    - Given a machine learning model the distance to the softball, calculated by the pixel radius of the detected softball, and the (x,y) coordinates, a machine learning model will guess the height.
    - After many iterations of training the model, we found it inconsistent and widely inaccurate.
    - The search for a geometric calculation technique was needed for fast and accurate return values.
- Known Radius Geometry
    - With the assumption that the radius stays constant for all softballs, we can use the number of pixels of the detected softball to find its distance from the camera
    - Given its (x,y) coordinates, we can determine the angle from the camera and use
- Pitching Line Calibration
    - The home plate and pitcher's mound are identified within the camera frame

    - The pixel-to-distance ratio can be calculated with the known distance between the bases.
    - The pixel-to-distance ratio was inaccurate as the vertical ratio of pixels differs from the horizontal ratio.
    - The ratio found in this method will only determine horizontal distances, not vertical heights.
    - The search for a vertical distance conversion is needed to find the height of a given (x,y) point.
- Known Heights Calibration
    - The home plate, pitcher's mound, and known heights are identified within the camera frame
    - The vertical pixel-to-distance ratio can be calculated from the home plate to the pitcher's mound line and known height's lines.
    - The height can be calculated given the (x,y), getting the number of pixels from the pitching line, and translating it into feet.
    - Used in the final design.

4.2.3 Decision-Making and Trade-Off

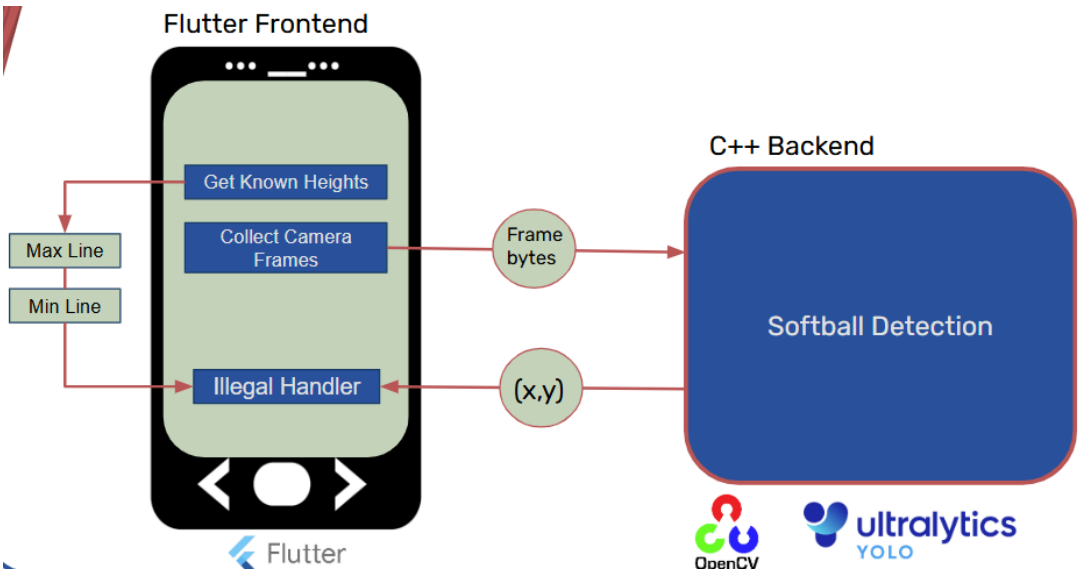
Design Option	User Compatibility	Observed Accuracy/Performance	Mathematically Supported
Multi-Camera Detection	**3/10** It would require users to place and have access to multiple cameras running the application	**9/10** Research shows this method is fairly accurate in triangulating a point, but the camera connection will delay the computation.	**10/10** Triangulation is a technique used in satellites for GPS tracking for its accuracy.
Machine Learning	**10/10** No setup would be needed as the machine learning model will handle the field differences.	**4/10** The machine learning algorithm struggled to accurately depict heights when the camera was moved into a new environment.	**5/10** Machine learning is the analysis of data trends, not a direct hard-coded mathematical relation between ball position and height.
Known Radius Geomtry	**9/10** The height of the camera would need to be inputted by the user.	**3/10** The radius of the ball becomes very small when the camera is placed far away, so the exact distance jumps when the radius increases by one pixel.	**4/10** The angle of the ball to the camera based on its position in the frame varies from camera to camera.

Pitching Line Calibration	**7/10** You would have to select 2 points on the camera feed as a part of the setup.	**8/10** The observed height looked fairly accurate when measuring a few trials in person, but reading larger heights prompted larger errors	**7/10** The horizontal pixel-to-distance conversation ratio is similar to vertical, but not quite the same.
Known Heights Calibration	**6/10** You would have to select 4 points on the camera feed as a part of the setup.	**9/10** The observed calculated heights seemed accurate, and the vertical distance distribution of pixels was visually equivalent.	**9/10** The vertical known heights are equidistant when viewed and can be used to calculate a strong vertical pixel-to-distance ratio

We chose the known heights calibration because of the mathematical accuracy it provides. The observed height still needs to be rigorously tested but has the most significant potential of the options. The multi-camera detection option was not considered because it did not align with our cost-effective user needs and requirements for fast computations.

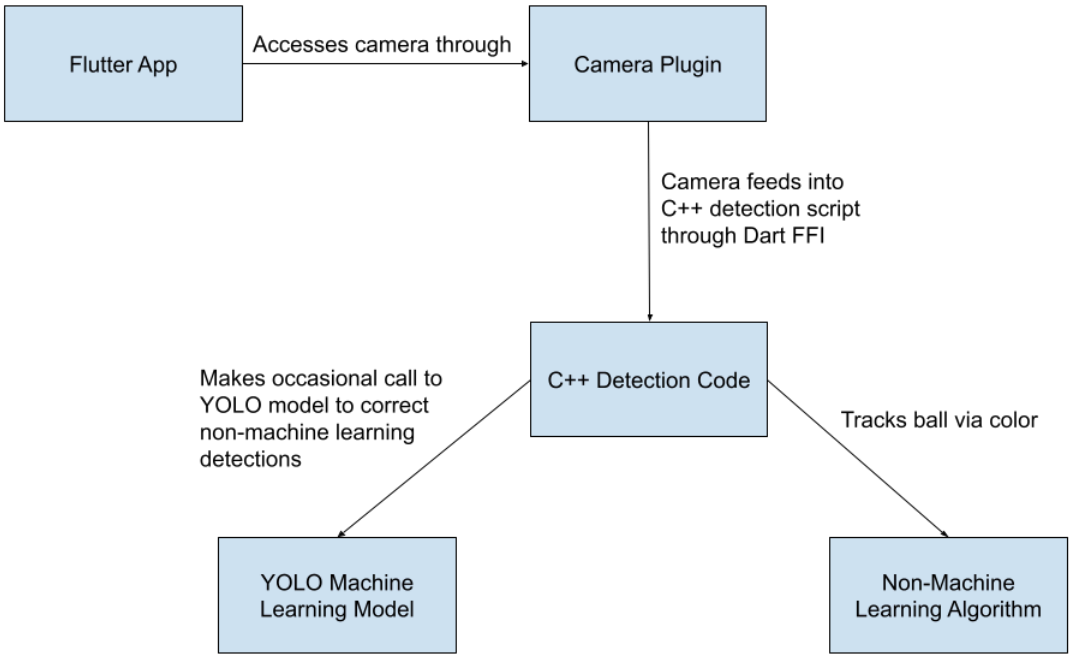
4.3 Proposed Design

4.3.1 Overview



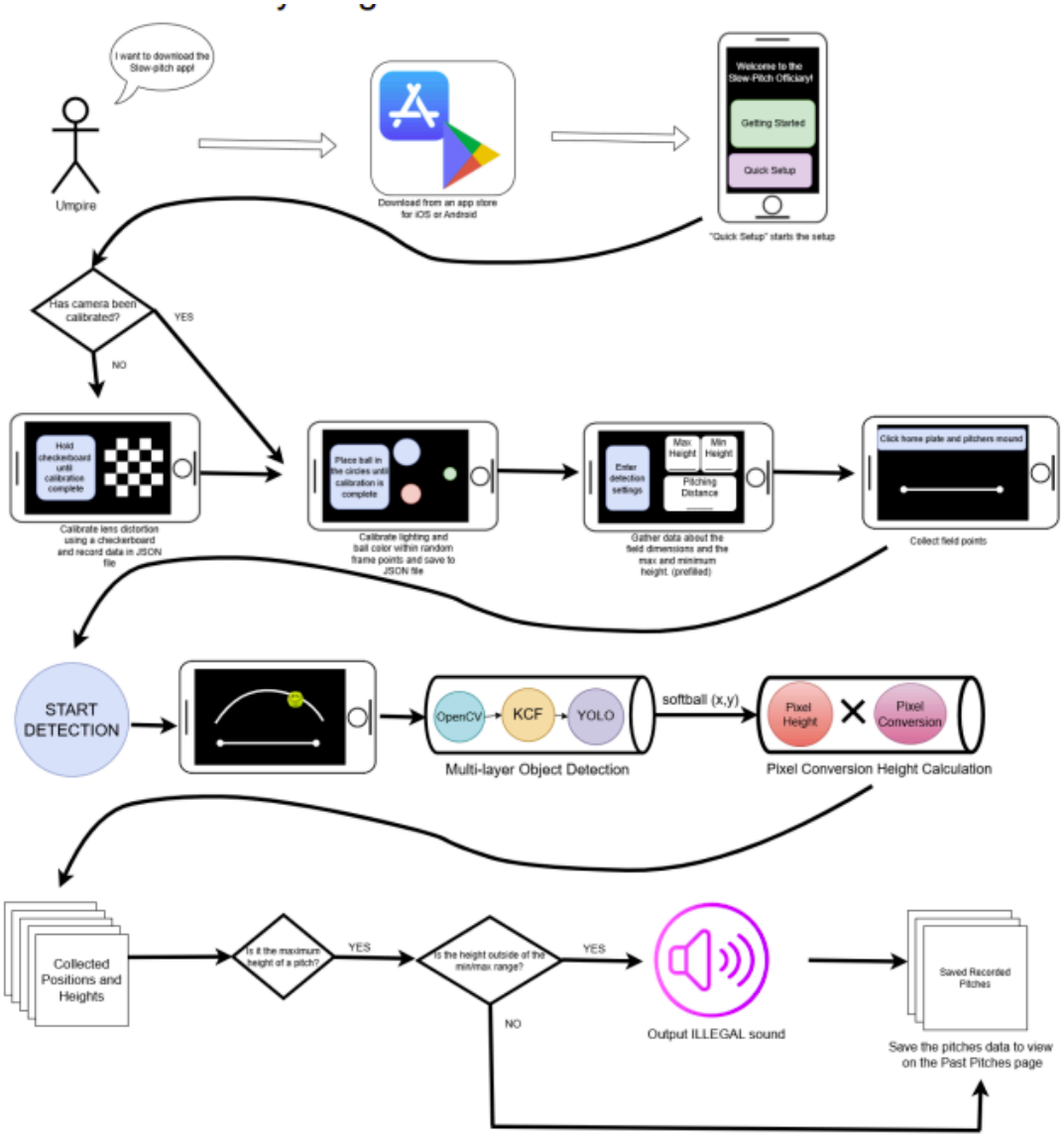
4.3.2 Detailed Design and Visual(s)

Starting with the Flutter app on a mobile device, users will have access to set up calibration values (distance from home to the pitcher's mound, ball color, etc.) for the camera to fit their specific environment. When they enter the tracking screen, our app will utilize a camera plugin to access their device's camera and begin feeding it into our C++ code using Dart FFI. Flutter is written in a code language called Dart. Dart FFI allows us to run C++ code in a Flutter environment. The C++ code will track the ball based on color for the majority of the tracking. Occasionally, the code will utilize a YOLO model. This model is created by essentially "training" the code to know what a softball looks like. This model will be used to correct errors in the color tracking as the environment variables change such as the lighting or changes of the ball's speed.



.

4.3.3 Functionality



4.3.4 Areas of Concern and Development

While our design is currently in development, the requirements generated by our users align with the trajectory of our progress. We are in the process of developing a fully-featured mobile application with the capabilities of processing softball pitches in real-time – the application is shaping to be easy to use with a clear prioritization of user experience, essential for ensuring both referees and players will be able to benefit from the use of the application. Our primary concerns as of now are ensuring the application runs at a reasonable rate with our softball detection solution, which could affect the times at which pitch calls are made. Our immediate plans for circumventing this problem is through thorough testing of our softball detection on native devices to ensure proper detection runtimes, as well as the continuous optimization of existing detection processes. As we move into fully integrating our solution onto a mobile device, potential questions for clients include what other additional features would improve the user experience of our application and what aspects of our current design could deter from the flow of a softball game?

4.4 Technology Considerations

Our project consists of multiple components that help integrate the two main components of our design: object detection and mobile app development. We use Google's app development tool Flutter, which can compile to run on an Apple iPhone, Android mobile device, or mobile emulators. Flutter uses Dart for an accessible widget-based design. Flutter's plugin functionality allows a smoother cross-platform development when utilizing device components such as the camera. Using Flutter, we can satisfy our technical challenges of developing an aesthetically pleasing and accessible user interface with a C++ backend.

Integrated into Flutter's backend, we use a combination of C++ libraries for object and motion tracking of the softball within a camera frame. There exist many sports ball tracking systems currently on the market. However, these systems typically guarantee accuracy with a multi-camera setup with expensive, high-quality cameras. Our design accommodates local league use with a camera on a mobile device. With a singular camera on the field providing one plane of vision, our application's accuracy relies on precisely detecting the softball within a camera frame. We do this using OpenCV's KCF algorithm, which detects an object by color and motion over time. Additionally, we are using YOLO's machine learning model to pinpoint the location of a softball; however, YOLO is more resource-intensive and is used as an infrequent correctional model to maintain accuracy. Integrating multiple object detection algorithms, libraries, and techniques, we can use the location of a ball within a calibrated camera field and quickly determine ball heights.

4.5 Design Analysis

So far, we have successfully integrated OpenCV and YOLO in C++ to create a framework capable of detecting and tracking a softball in flight. This integration has been efficient, allowing us to confidently believe that our proposed design from 4.3 will be feasible in practice. We currently have a working Flutter app, and we've been able to execute C++ code within the app on an iPhone. However, we are encountering a significant challenge: although our object detection and tracking code runs smoothly in the app on a desktop environment, it's not yet functioning on a mobile device. The issue stems from needing a custom OpenCV framework for iOS, and a bug in the most recent OpenCV version has prevented us from successfully building this framework. If we cannot

identify a workaround for this bug, we may need to consider using an earlier OpenCV version that does not present this issue. Moving forward, we have two main areas to focus on. First, we must resolve the OpenCV framework bug, as this is essential to achieving full functionality on iOS. Second, we must optimize our object detection and tracking code to reliably track the softball even when it's moving quickly. Currently, the algorithms struggle to capture the ball when it appears as a streak in each frame. To address this, we are considering increasing the frame rate; modern smartphones can support up to 240 fps, enabling us to capture higher-quality images of fast-moving objects. Our next steps involve continuing to troubleshoot the framework bug and exploring optimization techniques that can handle high-speed tracking. At this point, we don't foresee any major feasibility issues with our overall design—our challenges are primarily in the build and optimization stages. If we can overcome these, we expect the project to perform as designed

5 Testing

5.1 Unit Testing

Our solution is primarily software based; this implies the necessity for proper unit testing for each separate piece of functionality within our code. Because our main solution functionality is divided among two separate technical areas, the Flutter frontend and our tracking backend, unit testing is treated differently between the two implementations.

Within our frontend, unit testing occurs with each individual "widget", or visual component of our user interface to ensure items appear correctly on screen from our mobile application. This testing ensures that the user experience of our app remains consistent and that there is a high ease of access for users. In terms of the tools used for unit testing within our frontend, Flutter comes bundled with real-time debugging tools, useful for diagnosing and troubleshooting problems that occur when running an app. These tools can be accessed via a locally hosted interface accessible upon starting the Flutter app.

Within our backend, unit testing occurs on each individual method. This involves a process of ensuring, in all cases, the output from a certain method matches its expected functionality. This unit testing helps to prevent hard-to-trace issues that may populate further along in development.

5.2 Interface Testing

The interfaces in our design include the graphical user interface built from Flutter, the services included in the front end used to communicate with the backend, as well as the API included in the backend. Because our solution resides purely on a mobile phone, no external interfaces are necessary. The testing of the communication between interfaces will be conducted by ensuring that data is properly transferred between the front end and the back end; for the case of image data, for example, it is necessary to ensure that camera input is properly transferred to the correct image processing endpoint in our backend. The tools to be used with this type of testing include debugging libraries included in both our Flutter frontend via Dart, as well as debugging libraries included in C++. Xcode and Android Studio, the two IDEs responsible for building iOS and Android apps respectively also provide support for further interface testing which will be essential in the proper coverage of our solution.

5.3 Integration Testing

The critical integration paths in our design are ensuring that all front-end graphical components fit together seamlessly, as well as the front-end interface communication services are properly able to communicate with our backend image processing to retrieve modified image data. In reference to our requirements, these paths are critical as our solution needs to provide a seamless user experience while also ensuring our solution stays performant to the expectation that it can facilitate a real-time softball game.

These paths will be tested by connecting the elements of our solution and measuring the latency, unexpected visual artifacts, and general correctness of the resulting output. The tools we can use to facilitate this process include the aforementioned IDEs corresponding to iOS and Android development, as well as general debugging tools included in the Flutter SDK and in C++.

5.4 System Testing

System testing within our application involves testing the fully-integrated system in a real-world environment. Because our solution involves the physical tracking and processing of image data in relation to softball, much of our system testing occurs on softball fields, pitching softballs and recording the accuracy and speed of our application.

Specifically, in reference to the requirements of our solution, system testing must ensure that our solution is functional from a non-physically-obstructive point in the field of play and that various environmental factors (e.g. lighting) don't interfere with the functionality of our application. System testing also reinforces that our solutions accuracy in its height detection calls aligns with our requirements.

In reference to the tools required to complete such system testing, physical height indicators can be used to ensure a pitch call from our solution aligns with the actual height of the softball on the field. Other areas of functionality and their testing are covered in unit, interface, and integration testing.

5.5 Regression Testing

We have implemented a modular design strategy to ensure that each feature is separated in its implementation from other features; this strategy also fares well with other types of testing to provide an easy way to test certain features quickly and efficiently. By using this modular approach, it makes it very difficult to implement new features that break old ones; by separating our feature implementations, we ensure that each feature acts within its own scope without affecting the functionality of different features. This approach is driven by the requirement that our system remain efficient and that its ease of accessibility remains consistent.

5.6 Acceptance Testing

Our design requirements will be verified and validated through the proper testing of our system. Testing such as regression testing and system testing will help us to internally verify our requirements are being met. We will also cross reference our requirements findings with input from the client to ensure that our requirements are being met as much as possible; this cross referencing will help us to reduce bias from our standpoint by also weighing the perspective of our client. Input is to be gathered from our client through application demonstrations and progress reports.

5.7 Results

The results of our testing thus far are that the user experience of our system is operational, however the integration of our tracking solution needs further development to meet the standards of our users. Currently, the amount of latency present in our solution does not facilitate a real-time tracking solution per the requirements of our solution, defining the need for more testing to be done in terms of our interface design and integration.

For our next steps, as we continue to develop our solution and work on properly integrating our image tracking solution into our mobile application, we need to ensure that we focus on improving the latency of our application as well as continue to refine the user experience of our application to meet the standards of our users.

6 Implementation

Object Detection:

In our current iteration of softball tracking, we utilize C++, OpenCV KCF tracking, and Yolo models. The detection code is written in C++ and does the bulk of the tracking using OpenCV's KCF tracker, which detects the ball based on motion. Every 30 frames, a machine learning-based Yolo model trained on sports balls is called, which corrects any errors in the KCF tracking process. We only make occasional calls to the Yolo model to improve the overall runtime of our tracking code, which will help with the speed of the calls. We created a prototype of this implementation that draws a bounding circle around the ball on the screen. Still, the end solution will return the ball's position on the screen to compare to the height-reference lines we will draw on the screen for height detection.

Height Detection:

Our initial design idea was to measure the height of the ball from the camera and determine if the pitch is illegal; however, we were able to create an alternative solution by creating height lines on the screen based on images of known reference heights that will be taken by the user. With these height lines, we removed the need to measure the height of the ball. Now, we can utilize the

position of the ball on the screen return from the C++ object detection code and compare it to the position of the height lines to determine if the ball is out of the required height range.

Screen Development:

We have the main camera screen partially developed with access to record video from the camera and cache the recording in a history page. Accessing the from Flutter will allow us to pass individual frames to the object detection code. The video caching functionality will allow us to store past pitches for review or feedback once we fully integrate the height line and object detection implementation into our Flutter app.

7 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	Definition	IEEE Item	Team Interaction
Work Competence	Perform professional, high-quality work	“To seek, accept, and offer honest criticism	The team focused on this area by

		of technical work, to acknowledge and correct errors..."	developing solutions and experimenting with modern technologies to ensure we create the best solution.
Financial Responsibility	Deliver reliable products that are reasonably priced	"To be honest and realistic in stating claims or estimates based on available data..."	Our design choices are based on usability, such as having the app be free and not requiring an additional camera, which upholds this area.
Communication Honesty	Report truthful work to shareholders	"To be honest and realistic in stating claims or estimates based on available data..."	We have upheld this area by meeting weekly with the client and frequently communicating our progress and design choices.
Health, Safety, Well-Being	Minimize shareholders' health and safety risks	"To hold paramount the safety, health, and welfare of the public..."	The team wanted to minimize injury risk during the setup of our product, which helped lead us to our app-based solution, which eliminated the need to set up multiple cameras.
Property Ownership	Respect the property of others	"To credit properly the contributions of others..."	We followed this area by acknowledging the individual contributions of the team and by collaborating on different areas of the project so multiple members could gain knowledge in multiple areas.
Sustainability	Protect environmental and natural resources	"To strive to comply with ethical design and sustainable development practices..."	We focused on this area by making our app cross-compatible and usable on multiple iterations of

			phones, meaning our users will not need to continue to purchase new devices to use our product.
Social Responsibility	Make products that benefit society	"To improve the understanding by individuals and society..."	Our team has a responsibility to produce accurate data and to better the experience of rec league slowpitch softball players and umpires, so we focused on making our solution easy to use and available to slowpitch leagues.

Best Area of Responsibility; Social Responsibility:

We have chosen to approach this standard by designing our project to prioritize affordability, portability, and easy accessibility for local communities to have access to simple officiating assistance. Although this further complicates our software tracking model, it simplifies the application for the community. This decision in our development was ethically driven with the community in mind rather than solely for efficient development.

Worst Area of Responsibility; Sustainability:

We haven't been prioritizing the efficiency and runtime of our code up to this point as we're trying to get prototypes working, which could result in more power consumption and more contribution to negative environmental impacts. Our team plans on working to not only write code but to refine it to be as efficient as possible. Thus reducing the amount of power needed to run our app.

7.2 Four Principles

	Beneficence	Nonmaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	Our product aims to improve the state of rec league softball games, which benefits the enjoyment of everyone involved.	We are avoiding using multiple cameras which reduces risk during setup and improves the usability of our product.	We allow users to store past pitches, so they can make their own decisions based on non-subjective data, improving the enjoyment of the game.	Our app will provide unbiased, real data to help officiate games, which will improve the fairness of the game.

Global, cultural, and social	Promotes fair play by providing an objective tool for measuring pitch height, which will reduce arguments between players and officials.	Keeping our solution focused on a single device allows a normal game of softball to play normally without disrupting the existing league culture.	Since the height range of our app will be customizable, different cultures and leagues can set up the app to work for their needs.	The height range customization of our app will allow multiple leagues with different rules to improve the equality of how the game is called.
Environmental	Focusing on developing our project on a phone reduces the need for multiple tracking cameras benefitting the environmental impact.	Avoiding multiple cameras allows our users to use an existing phone instead of having to pay and ship a new camera.	Our product allows users to choose to use their existing phone instead of making more environmental impacts by paying for more cameras.	Our app will be able to be utilizied by all leagues, even ones with limited resources. This keeps them form needing to invest in other cameras that may contribute to environmental damage.
Economic	Our solution is free which eliminates the need to purchase expensive tracking equipment.	Using one device to perform all of the tracking helps us keep the app free because we can forgo the cost of an additional camera.	A league can make their own choices on whether or not to use our app or not, which gives them control over any economicals adjustments it my require such as a league phone to use the app.	The free price point of our app allows a wide range of leagues with different financial backgrounds can utilize our app to improve the league.

Area We Focused On; Public health, safety, and welfare X Beneficence:

We are aiming to improve the enjoyment of the game. We plan to achieve this by ensuring simple setup for umpires, returning accurate data so the right calls are constently being made, and being configurable for different height ranges so it can be used by a variety of leagues,

Area of Improvement; Environmental X Respect For Autonomy:

Based on the result of further testing, me may end up needed to restrict our app to only work on phones with a certain camera quality. Older phones with worse may not be able to provide the data needed for accurate tracking. The could result in leagues without immediate access to a high

quality phone having to purchase a new device through shipping methods that contribute to environmental damage.

We plan on overcoming this negative aspect by ensuring our app can run on a phones that are widely available and already owned by a majority of people. This will keep the availability of our app high and will reduce the need to purchase a new device.

7.3 Virtues

- Accuracy
    - We strive to provide accurate for our users, as it is important for illegal pitches to be called correctly as to not distress between players, officials, and fans.
    - We will achieve this virtue by continuously refine our ball tracking code and providing easy-to-use collibration steps so our app can be calibrated to fit the needs of each field.
- Empathy
    - Our app is solely focused on bettering the game for players, officials, and fans. We want to ensure that we are always prioritizing their needs over what makes it easier for us.
    - Communicating our design choices with our client and collecting feedback from testing are how we will ensure that we are always taking user needs into account.
- Innovation
    - We want our product to be easily integrating into existing leagues. So, we are innovating a single camera design approach that will make it easy to set up without the needs for multiple cameras.
    - Constant prototyping and research into different tracking methods will help keep us engaged with new ideas and strategies as to how to improve our single camera solution. Current team innovation like or height line system are already helping us improve our height detection methods.

Each team member should also answer the following:

- Identify one virtue you have demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - How have you demonstrated it?
- Identify one virtue that is important to you that you have not demonstrated in your senior design work thus far? (Individual)
    - Why is it important to you?
    - What might you do to demonstrate that virtue?

- Sully
    - Demonstrated Virtue: Persistence
        - With any project, there are bound to be roadblocks that come up that prevent progress. I believe it is essential that I am persistent in attacking these roadblocks and continuing to experiment so that we can solve our problems.

        - I demonstrated this virtue by continuing to attempt to integrate our C++ code into the Flutter app. There have been many issues with the integration, mainly stemming from the OpenCV libraries having issues running on iOS, and I need to persist through this issue so we can complete our app. While I haven't been able to fully integrate the code yet, I have made good progress through the OpenCV tracking framework for iOS I helped create.
    - Non-Demonstrated Virtue: Adaptability
        - We are utilizing many different technologies in the making of our app. So, it is important that I become familiar with the all of the technologies we are using, so I now how all of the components work together in detail.
        - I will demonstrate this virtue by pairing with other team members who are working on different areas and learning how they developed their ideas so I can help with issues in the future.
- Cameron
    - Versitility
        - This project has brought me outside of my comfort zone numerous times and required me to do things I have no experience with. Learning flutter and all of the ML topics was new to me and forced me to go outside my current knowledge to complete the tasks.
        - This is important to me because it is a necessity in the workforce. New technologies are constantly popping up, so a good engineer must be able to adapt to these new technologies and learn how to best use them.
    - Communication
        - I felt like this semester, I wasn't always communicating well with my team and that hurt my work. With better communication I would be able to work through issues quicker and help the engineering process as whole.
        - Hopefully, I'll be able to attend the weekly meetings next semester which will help immensely. Other than that, I just need to be more willing to reach out to my teammates for help and to keep them updated on my progress.

8 Closing Material

8.1 Conclusion

We have made individual components for the three main parts of our project including object tracking, determining the height of a pitch, and developing the main screen of our app. In the future, we need to fully integrate the components in Flutter and begin testing complete functionality. We also plan to continue refining our tracking model, improving the look of our app screens, and ensuring our height detecting methods are accurate across different environments.

The best plan of action to achieve our goals by condensing our C++ code to only focusing on taking a single frame and returning the the position of the ball. We want a majority of the code to exist in the Flutter side of our project which reduces the load on the back-end, which will improve the speed of our tracking algorithm. We will also continue to receive user feedback on the appearance of our app so we can provide the most user-friendly product we can.

Currently, we are experience issues integrating our C++ code into our Flutter application. In future iterations, we could focus more attention on getting our C++ integrated as our end product requires us to be able to execute our code on a mobile device.

8.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE). See link:
https://ieee-dataport.org/sites/default/files/analysis/27/IEEE%20Citation%20Guidelines.pdf

[1] “C Interop using Dart:FFI,” Dart, https://dart.dev/interop/c-interop (accessed Dec. 7, 2024).

[2] “Required packages,” OpenCV, https://docs.opencv.org/4.x/d5/da3/tutorial_ios_install.html (accessed Dec. 7, 2024).

[3] “Material library,” material library - Dart API, https://api.flutter.dev/flutter/material/material-library.html (accessed Dec. 7, 2024).

9 Team

9.1 Team Members

- Ethan Gruening
- Casey Gehling
- Sullivan Fair
- Josh Hyde
- Cameron Mesman
- Andrew Vick

9.2 Required Skill Sets for Your Project

- App Development
- Flutter Framework
- Machine Learning
- OpenCV
- C++

9.3 Skill Sets Covered by the Team

- Ethan Gruening:
- Casey Gehling:
- Sullivan Fair: App Development, Frameworks
- Josh Hyde:
- Cameron Mesman: App development, Flutter, C++
- Andrew Vick:

9.4 Project Management Style Adopted by the team

- Agile

9.5 Initial Project Management Roles

- **Team Organization**: Ethan Gruening
- **Client Interaction**: Casey Gehling
- **Machine Learning Integration**: Andrew Vick
- **Individual Component Development**: Sullivan Fair
- **Research**: Josh Hyde
- **Testing**: Cameron Mesman

9.6 Team Contract

Team Name _____**sdmay25-11**_____

Team Members:

1) _____Andrew Vick_____ 2) _____Casey Gehling_____

3) _____Cameron Mesman_____ 4) _____Joshua Hyde_____

5) _____Ethan Gruening_____ 6) _____Sullivan Fair_____

7) _____8) _____

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings:

1 pm every Wednesday (possibly)

2 pm every Monday with advisor (Dr. Fila)

Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Discord

Decision-making policy (e.g., consensus, majority vote):

Majority Vote

Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Andrew Vick will record meeting minutes, which will be stored in a document on our drive, and I will send a message in Discord for that meeting.

Participation Expectations

Expected individual attendance, punctuality, and participation at all team meetings:

We are expected to attend, share, and be on time for all meetings

Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Everyone is expected to meet deadlines, make meaningful contributions to the project, and be on time.

Expected level of communication with other team members:

Everyone is expected to be active in the discord and respond to teamwide questions within one business day.

Expected level of commitment to team decisions and tasks:

Everyone is expected to be committed to team decisions and tasks. This extends to actively engaging in discussions and conveying your ideas.

Leadership

Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

Team Organization: Ethan Gruening

Client Interaction: Casey Gehling

Machine Learning Integration: Andrew Vick

Individual Component Development: Sullivan Fair

Research: Josh Hyde

Testing: Cameron Mesman

Strategies for supporting and guiding the work of all team members:

Weekly standup meetings where we discuss the week's events and ensure everyone is keeping up with the workload.

Strategies for recognizing the contributions of all team members:

During weekly standup every member will share what they worked on and any information they found regarding the development of our product. This allows every member to share their contributions and receive feedback.

Collaboration and Inclusion

Andrew Vick, Software Engineering. Some of my relevant skills include C/C++, Python, and IoT devices. Most of my experience has come from personal projects for instance, using Python, I created my own virtual assistant.

Casey Gehling, Computer Engineering. Relevant technical skills are Embedded Programming, Networking, Fullstack Development, and UI Design. Additionally, I have relevant experience in client communication and requirements gathering.

Sullivan Fair, Software Engineering. Relevant skills include general coding knowledge of Java, C, C#, Python, JavaScript, AWS knowledge, GIT, and cybersecurity knowledge.

Ethan Gruening, Software Engineering. My relevant skills include, but are not limited to, Python, C, Java, JavaScipt, and AWS. Most of my projects/experiences involve user interphases, I/O devices, and API service integrations.

Josh Hyde, Computer Engineering. My relevant skills are generic coding in C/C++ and Java, and some database work with MySQL. I also have been in different groups before and have had to work together towards a common project goal.

Cameron Mesman, Computer Engineer. My relevant skills are coding with C and Java. I have experience programming embedded systems, app design (backend, frontend, and database languages), and computer architecture.

Strategies for encouraging and supporting contributions and ideas from all team members:

Goal-Setting, Planning, and Execution

Team goals for this semester:

Formulate an initial product design.

Have an action plan for developing the product.

Strategies for planning and assigning individual and teamwork:

During team meetings, we will identify what needs to be worked on for the week. From there, we will assign who tackles which task based on their expertise.

Strategies for keeping on task:

Weekly goals and check-ins

Consequences for Not Adhering to Team Contract

How will you handle infractions of any of the obligations of this team contract?

In the event of an infraction by a team member, the team as a whole will reach out to the member to see why they aren't adhering to our contract.

What will your team do if the infractions continue?

If the infractions continue and we cannot resolve the issue internally, we will finally reach out to Professor Fila regarding the member.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the

consequences as stated in this contract.

1) _____Sullivan Fair_____ DATE __09/18/2024_____

2) _____Andrew Vick_____ DATE __09/18/2024_____

3) _____Casey Gehling_____ DATE __09/19/2024_____

4) _____Ethan Gruening_____ DATE __09/19/2024_____

5) _____Josh Hyde_____ DATE __09/19/2024_____

6) _____Cameron Mesman_____ DATE __09/19/2024_____