Design Document #4

SDMAY25-11

4.2 DESIGN EXPLORATION

4.2.1 Design Decisions

CREATE AN APP THAT RUNS THROUGH A PHONE CAMERA

We decided to take an app-based route because we want our solution to be accessible to all types of users. While current ball tracking solutions may provide super fast and accurate tracking, such as the Goal-Line system in soccer or the Hawkeye camera in tennis, these rely on multiple cameras and expensive equipment to triangulate the ball's position. Our solution is based on recreational softball. So, a majority of users will not have access or the desire to obtain those resources. A simple app that is easy to set up and use, and doesn't require large financial commitments is what we aim to create.

UTILIZE THE FLUTTER FRAMEWORK FOR APP DEVELOPMENT

One of the issues with making an app is how to develop it for a specific platform. We didn't want to restrict our product to a certain platform, as that would prevent users on the opposite platform from using the app. However, developing two different code bases to accommodate both platforms would have caused much more work for the team. So, we decided to utilize Google's Flutter framework which allows iOS and Android development to exist in the same code base. This framework allows us to develop both sides at one time, which lets us spend more time refining and improving our tracking algorithms and UI design.

USE A HYBRID MACHINE/NON-MACHINE LEARNING DETECTION METHOD

When deciding on what tracking method to use, our initial approach was to use only machine learning. This idea came because we wanted to train a model that accurately tracked softballs in various lighting conditions as our product will be used during the day and night. This led to design issues because machine learning is slower we needed to be able to call illegal pitches at least as fast as a normal umpire. With this in mind, we decided to look into non-machine learning, and while it didn't handle changes in environments well, we found that with some calibration, we could get some pretty accurate and fast detections. So, we wanted to find a way to implement the speed of non-machine learning with the accuracy of non-machine learning. Our current method now involves using non-machine, color-based detection to pick up the ball

initially. Then, we can pipe the camera feed to a YOLO model after a certain number of frames to correct any error in the non-machine learning detection. This method will make our detection more efficient while not compromising on accuracy.

4.2.2 Ideation

One of our design decisions was how to (detect the softball, the use of an app, height detection) One of our design decisions was how to accurately and consistently identify the softball. We though of many different solutions to this problem. We considered using a color-based non-machine learning solution that could detect the ball using just the color of the ball. Additionally, we considered using a machine learning approach where we manually fed in data for the machine learning to learn the softball. We also considered using color-based object detection and some kind of motion-based detection to identify the ball more accurately. Another solution we considered was to incorporate a machine learning-based solution that took into account the YOLO model's already pre-trained data of different kinds of objects including sports balls. The final design option that was considered was a sort of middle-ground approach that would use non-machine learning object detection techniques such as color-based object detection, on top of also using a machine learning-based approach that would use pre-trained models.

4.2.3 Decision-Making and Trade-Off

1. <u>COLOR-BASED DETECTION MODEL</u>
    - We identified some of the pros and cons of this solution by implementing and testing our own models that incorporated color-based detection models. We also did some research as well to test the reliability of just the color-based detection model.
    - Pros: Very efficient and fast, very accurate up close.
    - Cons: ineffective at farther distances, very inconsistent in differing environments and brightness.
2. <u>MANUALLY INPUTTED DATA FOR MACHINE LEARNING</u>
    - We identified some of the pros and cons of this by manually inputting some of the locations of where a softball was with hundreds of different images to train the model to start to learn what the softball is and looks like. This seemed like a good idea until we found the models that already exist that are pre-trained on certain objects like the YOLO model.
    - Pros: gets more accurate with more data, better at identifying softball in differing environments.
    - Cons: Very tedious and time-consuming, not guaranteed to work.
3. <u>COLOR AND MOTION-BASED OBJECT DETECTION</u>

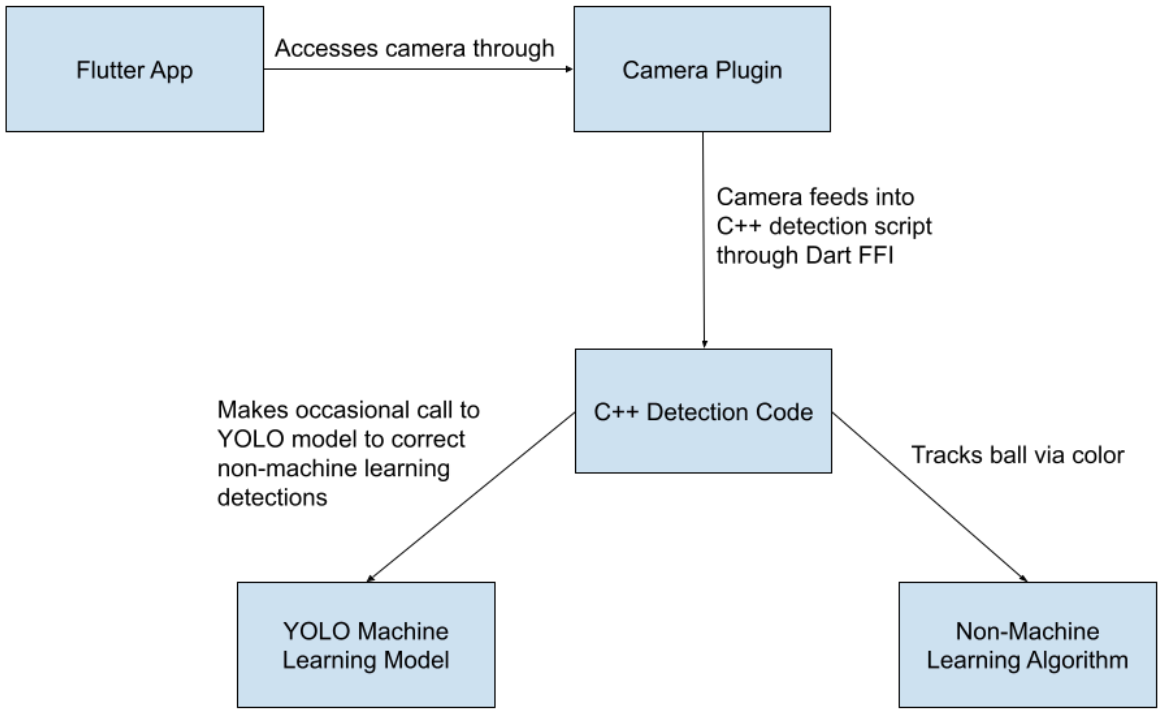
- We also tried implementing a non-machine learning scenario that involved different types of ways and inputs in identifying the softball. We also researched some different non-machine learning ways to identify an object, and one of them is based on motion, but with other objects moving and or an unsteady camera, this way isn't as reliable without the color-based as well, however, it didn't really improve anything at all.
- Pros: Very fast and easy to implement, good up close
- Cons: bad at different environments, not very accurate or reliable at longer distances.

4. MACHINE LEARNING USING PRE-TRAINED MODELS
    - We also identified another machine learning solution that involves already pre-trained data so we don't have to manually teach an AI what a softball looks like. We researched different models and the YOLO model seemed like it could work the best because it had a sports ball object that it was trained with that could consistently identify sports balls.
    - Pros: More accurate, not too difficult to implement, better at different environments.
    - Cons: Still could be improved, not as accurate as we want, slow
5. COLOR AND MACHINE LEARNING MODEL-BASED DETECTION
    - We considered using a mix of machine and non-machine learning to make a model that could more accurately identify the ball trying to use the pros of the respective aspects of the other solutions. We identified the pros of the non-machine learning color-based detection in being fast and pretty accurately just identifying the ball up close, so we have color detection as the primary initial part in identifying the ball. We also have the machine learning aspect with the model to more accurately identify the ball not as frequently to make the solution not as slow.
    - Pros: Most accurate that we have identified, better at changing environments, not too slow.
    - Cons: Worse at farther distances and certain lighting.
    - We decided to choose this approach as the best approach moving forward as if implemented accurately and correctly, would be the best at identifying the object consistently and better in changing or differing environments. It takes the benefits of non-machine learning and machine learning and tries to incorporate only the benefits from both. It serves as a middle ground between effectiveness and efficiency as the solution needs to be accurate and quicker than the speed of a normal umpire.

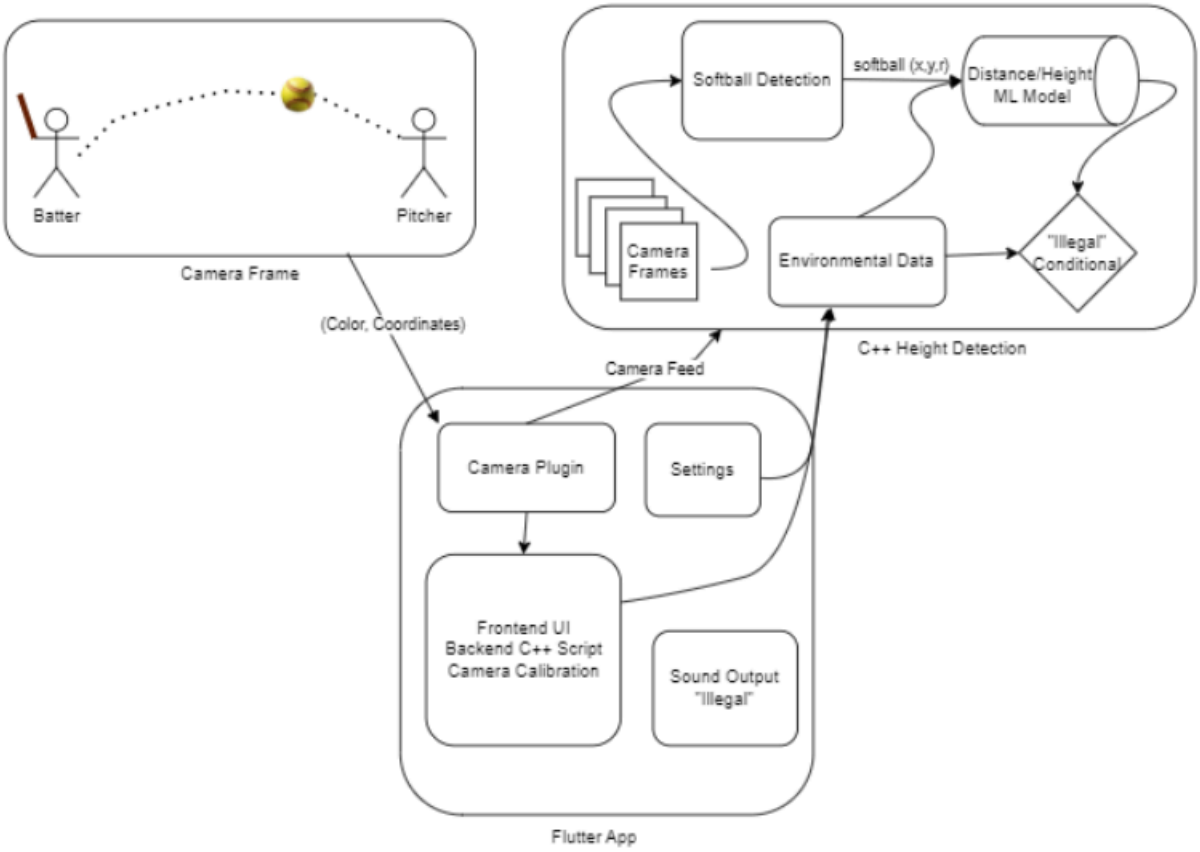
4.3 PROPOSED DESIGN

4.3.1 Overview

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.



4.3.2 Detailed Design and Visual(s)



Our softball detection application is built for the purpose of detecting a softball during a pitch to determine if a pitch falls within the legal height boundaries as dictated by league rules. The components of our system include:

- A mobile application built with Flutter providing a user interface with a real-time display from our softball detection solution
    - The user interface must be updated in real-time during the gameplay, and passed pitch decisions must be available.
- An object detection system capable of identifying the position of a softball relative to a camera
    - OpenCV is used in tandem with preexisting open-source sportsball detection models to identify softball positions from a video input consistently.

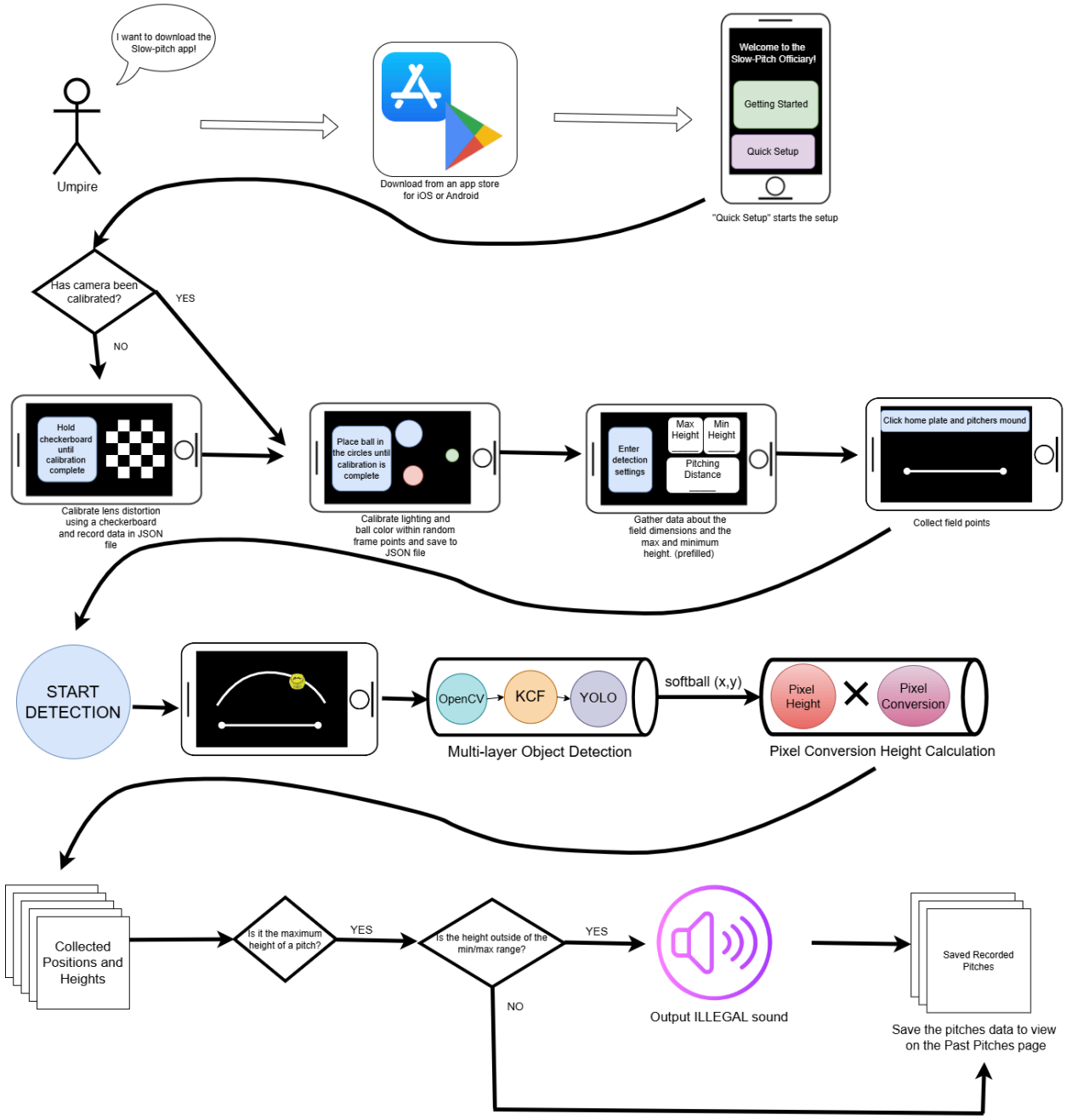
Requirements

- Must be able to identify an illegal pitch correctly
- Decisions must be just as fast or faster than an average umpire (at least within .3 seconds of max height)

- It must not interfere with the field of play
- It must be identifiable and clearly indicate to the batter whether the result of the pitch is illegal or legal

The app and the object detection are used together to simultaneously detect the legality of the pitch and convey the result to the batter.

4.3.3 Functionality Diagram



Above is how the user would instructional download our application, run, and set up an additional officiating tool during their slow-pitch softball game.

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 design uses OpenCV, YOLOv5, and the Flutter framework for capturing, detecting, and tracking a moving softball.

- **OpenCV**: Handles image processing and tracking. It's fast and versatile but requires a custom-built framework for iOS, and we're encountering a bug with the latest version that may force us to use an older release or explore alternatives like Apple's Vision framework.
- **YOLOv5**: Provides efficient object detection, balancing speed and accuracy for real-time tracking. However, it's focused on detection rather than tracking, so we've had to integrate separate tracking algorithms. We may explore lighter versions (like Tiny YOLO) if more optimization is needed.
- **Flutter**: Enables cross-platform deployment and smooth UI design. While it allows us to deploy on both iOS and Android, integrating it with our C++ code on iOS is challenging, and further issues could lead us to consider a native iOS app.

Overall, each technology was chosen for its ability to handle real-time tracking. Our main focus moving forward is resolving the iOS integration issues and optimizing performance for fast-moving objects.

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.