Radminton

Decision Support in Racket Games

Design Document

sdmay20-44
Client: Simanta Mitra
Adviser: Simanta Mitra

Team
Benjamin Kramer - Meeting Facilitator
Brian Guidarini - Test Manager
Katelyn Sinn - Schedule Manager
John Rachid - Delivery Manager
Aiden McMinimy - QA Manager
Christion Barnes - Report Manager

Executive Summary

Development Standards & Practices

Software Practices

- Agile - The team uses Agile practices like iterative sprints, delivering working software within each iteration, communicating with the client, and pair programming.
- Refactoring - The team is dedicated to refactoring at regular intervals to ensure the health, reliability, and readability of the codebase.

Engineering Standards

- PEP 8 Style Guide for Python Code [5]. Since this project is primarily developed in Python, the team became familiar with the PEP 8 Style Guide for Python Code to ensure our code followed industry standards and stayed readable.

Requirements

- Videos of badminton games can be analyzed to generate feedback for users.
- Livestream video of badminton match can be analyzed to provide real-time audio feedback to players.
- Users must provide a video they recorded for non-live games or set up a camera of their own for live capture.
- All video analysis can be done on a computer via our website.

Applicable Courses from Iowa State University Curriculum

In our time at Iowa State University, we've been lucky to have opportunities to learn a lot of important concepts that made this project much easier on us. Without these foundational courses, this project would likely not be achievable in the time given to us.

COM S 227: Intro to Object-Oriented Programming

- Taught team members object-oriented programming, a valuable asset in this project.

COM S 228: Intro to Data Structures

- Taught team members about data structures, a vital subject for all programmers.

COM S 309: Software Development Practices

- This course was the first major project course for most of the team and helped us develop as team members and developers.

SE 329: Software Project Management

- In this course, we learned the basics of project management, helping us to keep our team focused, on time, and even make this document.

COM S 472: Principles of Artificial Intelligence

- This course gave a couple of members of the team a framework for thinking about intelligence in software. With this knowledge, more sophisticated solutions to our decision engine were able to be found.

MATH 207: Matrices and Linear Algebra

- Linear algebra is highly relevant to the project, as our computations and visualizations rely on an understanding of mathematical transformations.

CPR E 575: Computational Perception

- This course taught many team members about computer vision and perception. This course helped give the team a jumpstart by shortening the time it took to learn new technologies.

New Skills/Knowledge acquired that was not taught in courses

Computer Vision/OpenCV

- While some team members were already familiar with these concepts, half the team was not. This gave that half a chance to learn these topics without taking a formal course on the subject.

Python

- Not all of the team had been exposed to Python before this project. Now, all members know how to write quality Python code.

Badminton

- Our team was not very familiar with the sport of badminton before this project, so we took the time to learn how to play the game to give us better insight into how the application should function.

Project Goals and Deliverables

Goal	Description
Player tracking	Player locations are found in screen space.
Ball Tracking	The ball location is found in screen space.
Court Identification	The court boundaries are found in screen space.
Player Location	The location of the players are found in world space.
Ball Location	The location of the ball is found in world space
Court Location	A 2D representation is created of the court.
Ball Trajectory	The trajectory of the ball is calculated.
Player Trajectory	The trajectory of the players is found.
Player Location Suggestion	An ideal location is suggested for the player.
Stroke Suggestion	A stroke type (backhand, overhead, underarm, etc.) is suggested for the player.
Return Location Suggestion	An ideal area to hit the ball towards is suggested.
Give feedback	The application suggests ways the player can improve.

Deliverable	Description
Video Analysis	Given a video, our application can give ball trajectory, player trajectory, locations and more.
Feedback from Video	Given a video of gameplay, our application can give suggestions on how to improve.
Real-Time Feedback	Given a live stream of video, our application can suggest ideal player actions in real time.
Vocal Communication	Our application can give audio suggestions over a Bluetooth earpiece.
Website	The project will feature an easy-to-use website as a means to interact with the software.
Web Service	The project needs a way to support the use of the website, delivering all the functionality as a service.

Table of Contents

1 Introduction

1.1 Acknowledgement

The team would like to acknowledge our client and advisor Dr. Simanta Mitra who has provided project guidance, server resources and even badminton equipment to help the team learn the sport and come up with ideas.

1.2 Problem and Project Statement

Problem Statement

Badminton is a popular racket sport in which players hit a shuttlecock back and forth between a net until one player cannot return the shuttlecock. Despite its popularity, the sport does not have much in the way of sports technology to help casual players learn the game and competitive players improve. To remedy this, Dr. Simanta Mitra tasked us with finding a way to objectively analyze his badminton play so that he, and any other user, can figure out when they make mistakes and what they could have done better in those moments.

Solution Approach

Our project*, Radminton,* attempts to provide an easy-to-use but powerful tool to help competitive and casual players alike improve their Badminton game. Given a pre-recorded video of a badminton game, *Radminton* will be able to analyze and provide feedback to the user. On a hit by hit basis, users will be able to receive advice on where they should have been if they were unable to return the shuttlecock, where they should have hit the shuttlecock to maximize their chances of scoring a point, what kind of stroke type they should use and more. *Radminton* will eventually be able to provide suggested moves during a live game using Bluetooth earpieces and live streams of camera games.

1.3 Operational Environment

Radminton is expected to operate anywhere a modern smartphone or laptop is capable. However, if the user wants to record a game or get live suggestions, they must be at a badminton court. This is because our application relies on the markings of a badminton court to offer suggestions. Luckily, badminton courts are not hazardous, and even if the environment was a professional game, the large crowds and loud noises would not be an issue for *Radminton*.

1.4 Requirements

Functional

- Live streams of badminton games must be able to be processed in real time. This includes reading, analyzing, and suggesting each frame in a sufficiently small amount of time.
- The user must be able to hear the advice from the program through a Bluetooth earpiece.
- Pre-recorded videos of badminton must be able to be analyzed by our application.

Environmental

- The decision engine must be able to run off a laptop set up near the court.
- The software must be accessible and usable from a website using a pre-recorded video.

Economic

- Server space is being provided to us by the university.
- Bluetooth equipment and cameras will be to be provided to us by the university once this point in development is reached.
- Laptops are the primary driver of the project, and all members own one, saving significant resource costs.

UI

- The UI must show where players were at specific points in the rally.
- The UI must advise the player on how they could have improved, including but not limited to:
    - ■ Where the user should have returned the shuttlecock.
    - ■ Where the user should have been before returning/dropping the ball.
    - ■ The optimal type of return i.e. backhand, forehand, spike, etc.

1.5 Intended Users and Uses

Use Case Number	**Use Case**

UC-1	Get basic advice such as where to return a shot and where to move to be able to return the ball in real time.
UC-2	Use a pre-recorded video with the application to receive advice on where the user should have been or returned the shuttlecock.
UC-3	Get diagrams showing play by play interactions.
UC-4	Use a pre-recorded video with the application to receive advanced advice such as optimal stroke type and stroke strength.
UC-5	Access the application's website to upload videos and receive information.

Table 1

User	**Use Cases**
Novice Badminton Player	UC-1, UC-2, UC-5
Experienced Badminton Player	UC-1, UC-2, UC-3, UC-4, UC-5
Badminton Coach	UC-2, UC-3, UC-5
Badminton Spectator	UC-3
Sport Analyst	UC-3, UC-5

Table 2

1.6 Assumptions and Limitations

Radminton is being developed with the following assumptions and limitations in mind:

Assumptions

- The maximum number of players on the court is four (two players per team).
    - Typical badminton games do not exceed this number, so while we can confidently say any legitimate game will follow this assumption, users may have issues if they try to use it in ways that break this assumption.
- The shuttlecock is green
    - Shuttlecocks are usually white, but often they are green. Our application relies on color to determine where the shuttlecock is located, so if a user used a different color, they may have issues with the application.
- A user will be able to record the full court.
    - *Radminton* requires that the entire court be visible. This may not always be possible for the user.
- The court follows typical badminton dimensions.
    - While most courts follow the same standard court size, some courts may be unconventional, which would be a problem for our modeling and tracking.

Limitations

- Processing and analyzing a video should not exceed the length of the video itself.
    - For instance, a minute-long video should not take longer than one minute to process and analyze. Results should be given to the user within that time.
- The application shall be useable from a web browser.
    - Whether the user is using a phone or a laptop, the application should run through the browser.
- Tests have not been performed for all parts of the application.
    - While some test cases do exist, the vast majority of the application has yet to receive unit testing. A fair amount of quality assurance has been done, however.
- Access to skilled badminton players is limited.
    - While the team's advisor and client is a skilled badminton player, *Radminton* will not be able to be tested with professional players, limiting user testing.

1.7 Expected End Product and Deliverables

End Product

The end product of this project is a web application called *Radminton*, where users can upload videos of badminton games they've played and receive suggestions on ways to improve. The user can also connect a live stream to the application and receive badminton advice in real time through Bluetooth headphones. In both cases, *Radminton* can give back hit by hit breakdowns that include where the ball went, where the players were, and what the score was at any given time.

User Deliverable	Description
Suggestion	Given a video of gameplay, our application can give suggestions on how to improve including recommended stroke type, return location, and where the player could have improved. This functionality will be delivered on March 8th.
Real-Time Feedback	Given a live stream of video, our application can suggest ideal player actions in real time. This will utilize Bluetooth headphones to give players audio feedback. This feature is not expected to be delivered within this project cycle, and will more likely be completed within the next two years.
Website	To give access to the application to our users, we need a website to accept videos and send back information. This piece of the project is expected to be delivered by April 26th.

Table 3

2. Specifications and Analysis

2.1 Proposed Design

Our project can be broken up by its tasks; tracking, real-world locations, suggestions, and our website. For tracking, much of what we did implement OpenCV's background subtraction methods [4]. For players, we found the objects on the courts that were moving and filtered by size so as not to include the moving shuttlecock. Much the same with the shuttlecock, we used background subtraction but, due to its small size, we implemented another layer which was color filtering, which is why it is required for the shuttlecock to be a bright green. Due to its small size, just the movement was not enough to remove random movement elsewhere or false objects. Court detection uses OpenCV's contour finding methods [3]. Contours were then filtered out by length and then color, in our video, the color red. Since a requirement we have is live video analysis, all our algorithms need to be efficient enough to give the players feedback in a timely fashion. Therefore, we had to refactor and add improvements as we went.

Next, we had to translate that data from pixel coordinates to real-life locations. Since we knew the dimensions of a badminton court and we knew the pixel coordinates of the court, you can apply a homography translation [1] which OpenCV also has methods for. This is often used to take tilted flat surfaces and create a top-down view of them in a different reference frame. You can implement that here because you know the actual reference frame of the court and the pixel coordinates. The found homography matrix can then be applied to the bottom point of the player objects since their feet are on the same plane as the court.

While the player locations were somewhat simple because it was 2D, the ball location is a whole different ball game. A way to find the real 3D coordinates is by using two cameras and applying OpenCVs reference functions to it. Another way is by possibly curve fitting a line to predict where the shuttlecock will go. If we don't want to do real-time, we could simply "look ahead" to see where the shuttlecock lands or is hit to generate feedback for a user with an already recorded video. Since one of our requirements is real-time feedback, that would not be an option.

Suggestions are generated by using our real-world locations of the ball and players. We provide a suggestion on where the player should move to by finding where the shuttlecock is going. We provide a return location by choosing the spot where the opposing side is farthest from. We provide a stroke suggestion by seeing where the player is moving to hit the shuttlecock and what side the shuttlecock would be on with the least amount of distance to move.

Our website's design has been discussed minimally with our client, however, the plan is to save match data for a user so it can, over time, tell them their consistent weaknesses and strengths, save their matches, or show them better ways to play in a match. This data can be saved in a database for a user.

2.2 Design Analysis

The pieces we have working confidently are ball tracking, player tracking, court detection, real-world player coordinates, and return location. The frames per second is still slightly too slow to do in real time but we are always improving and is a focus of modification in our second semester. Background subtraction and color filtering for all tracking as mentioned in 2.1 worked as expected. Due to the difficulty of predicting where the shuttlecock is going, we have yet to find the real-world ball data or where to suggest where the player should move to. If we cannot find a good way to predict, we will have to fall back on just doing pre-recorded videos and use the “look ahead” method discussed in 2.1 for the real-world shuttlecock location.

Some strengths in our proposed solution are our knowledge of OpenCV documentation and what can be applied from it. In the case of homography transformation or background subtraction, we spent a lot of time researching our plans and what would fit best with the highest efficiency. Some weaknesses, however, are our knowledge on how to do 3D transformations and how color filtering removes the ability for many courts or shuttlecock colors to be viable which could be an improvement later on to allow more colors or not filter at all.

2.3 Development Process

Due to the structure of the class and how our team meets, we decided that Agile was our best option for our development process. Every week, we met with our client, Dr. Mitra, and discussed our updates, concerns, and thoughts on the project and our ideas. Additionally, we met as a team every week to figure out what things were working for us, how we could improve what we already completed, and our next goals. This is what our sprints looked like during this semester, with new developments happening about every week and a half to two weeks. Another integral part of our sprints was discussing and thinking about how our new goals would coincide with the current architecture of our project and if it could be easily integrated. We began to do this after we realized, at some point this semester, that our project code wasn’t really modular and as efficient as it could be. After remodeling our project architecture, we discuss as a group how we are going to integrate new code into our project, and how it may affect old and new modules in our framework.

2.4 Design Plan

Now that we have a working project, our future sights are set on implementing our suggestion engine, the real-time feedback mode, and then finally integrating all modules seamlessly. We will start this second iteration by developing and integrating the suggestion engine. This is the last module of our project and is arguably the most important.

For us to develop the suggestion engine, we need to extrapolate information from the data we are receiving through our player identification and shuttle identification classes. For example, for use cases 1 and 2, we need to be able to advise the user on where they should have been to gain the biggest advantage, based on their opponent’s player location data and the shuttlecock’s trajectory.

Once the suggestion engine is optimized for accuracy and is fully integrated with the rest of the project, we will finally be able to move to the final phase of the project: the website application. The website and its features are what the user will primarily be interacting with. For example, most users will be able to access the website and upload some type of video of badminton gameplay and get some information or feedback in return. After finishing the website, our final goal is to optimize the framework as much as possible to increase processing speeds, user interactivity, and finally improving UI aesthetics. Once all of these goals have been met, and last but not least, the client is satisfied with the product, our team's project will have been successfully completed.

3. Statement of Work

3.1 Previous Work And Literature

Previous work and literature related to this project includes a Badminton Game analysis app. This app allows for manual input of the results of badminton rallies. This can help with the analysis of badminton players or games. From the manual input, it can provide information like statistics or scatterplots of where the birdie was hit. One large problem with this app is everything is manual. We differentiate yourself from this work by planning on automating this input and providing live feedback while the game is being played. Another piece of related work is Fishified's Tracker3D Direct to Linear Transform [2]. This is an example of providing 3D coordinates for a moving object when using a normal digital camera. With this, you need to provide many details about the camera calibration and do some linear algebra to get the 3D coordinates. This calibration information must be accurate and cannot be easily found. In the end, we decided on not using this approach in order to provide the easiest experience for the user. This results in not being able to track the birdie as accurately.

3.2 Technology Considerations

We had various technology considerations when working on this project. The main consideration was using a normal digital camera (one that can be found on a phone) as opposed to a stereo camera. A stereo camera is not as accessible and more expensive but can provide accurate 3D coordinates for tracking a moving object. In the end, we decided on using a Digital camera for its ease of use and accessibility. This does, however, make it harder to accurately track the 3D coordinates of the birdie. This results in us having to put far more work into the identification Task[3.3]. Another technology consideration was Python Vs. C++. We knew we had to choose one of these languages since they were supported by OpenCV. C++ is a compiled language that would allow for faster computing. Although C++ is faster it does not outweigh the cons of the language. Some of these cons are that C++ is difficult to use and difficulty of importing other libraries. In the end, we decided to go with Python. Starting with the pros, Python is a very easy language to use and learn. Python also allows for easy use of other libraries. Python, however, is notorious for being difficult to debug in large codebases and is far slower than C++ as it is a scripting language.
Python Vs. C++. In the end, we selected Python due to its ease of use and ability to easily incorporate other libraries. Being able to use other libraries easily can make a very taxing problem into a quite simple one. This saves a large amount of time throughout this project.

3.3 Task Decomposition

Identification - Locating the coordinates of each crucial aspect of gameplay. These coordinates are required to be accurate as they will be used heavily throughout the project.

- Identify and return court coordinates
  - Automatically provide the coordinates of every line on the court,
- Identify and track birdie while providing the current coordinates
  - Provide the 3D coordinates of the birdie and update these coordinates every time the birdie changes position.
- Identify and track the players while returning the coordinates at different parts of their bodies
  - Provide the 2D coordinates of the players and update these coordinates every time either player changes their location.

Prediction engine - Using the identification data to give the player the optimal moves they should be making.

- Predict where the best location to return the ball is
  - This will recommend returning the birdie to the location fathers away from the defending player.
- Predict the best location to move to return the shot is
  - This will allow the defending player to be in the most optimal location to return the estimated birdie landing location.
- Predict the type of stroke you should return the birdie with
  - This will provide the defending player with the optimal stroke they should use in order to return the birdie.

Suggestion - Provide live suggestions and generate feedback to help the player become a better player.

- Generate feedback
  - This will inform the player or players where they could have improved on throughout the game. This will be able to be completed offline and is not in real time.
- Send prediction to player
  - This sends the data from the prediction engine to the player via Bluetooth. This is done in real time and will be used to tell the player the optimal decision.

3.4 Possible Risks And Risk Management

Some possible risks of this project are our group members course load, Players being given bad suggestions/feedback, and not being able to meet time frames. The course load is a risk since our schedules as students can vary. If our course load is high it can make it quite challenging to give this project the proper amount of time it requires. This can result in an inferior final project. Players being given bad suggestions or feedback could result in the user being less satisfied in our project. We plan to manage this risk by meeting at times that work for all group members and being understanding and helpful if a group member is unable to finish their part. Since our program analyzes the player's gameplay and provides feedback any bug or unintended behavior could give poor feedback. We plan on managing this risk by doing heavy unit testing and having the feedback be reviewed by competitive badminton players. The last risk is not being able to meet time frames. If we are not able to stay on schedule for this project we might need to cut things out of this project. This can also result in an inferior project. This is managed through the delivery manager and the use of a Gantt chart for scheduling.

3.5 Project Proposed Milestones and Evaluation Criteria

Our milestones are the completion of our three main tasks[3.3]. These milestones are as follows: completion of the identification task, completion of the prediction engine task and completion of the suggestion task. To evaluate that these tasks are completed, we will be using comprehensive unit tests. These tests will confirm that each task is returning accurate and proper information. Another evaluation we will use is review from competitive badminton players. These players will ensure that the optimal move is being returned each time and the program is working as intended

3.6 Project Tracking Procedures

We are using Trello for project tracking. This allows us to easily divide work up by iteration and easy assignment of subtasks to different members. Trello also helps iteration requirements to be more clear and prevents iteration requirements from not being completed.

3.7 Expected Results and Validation

The desired outcome is the creation of a product titled *Radminton*. *Radminton* attempts to provide an easy-to-use but powerful tool to help competitive and casual players alike improve their Badminton game. Given a pre-recorded video of a badminton game, *Radminton* will be able to analyze and provide feedback to the user. On a hit by hit basis, users will be able to receive advice on where they should have been if they were unable to return the shuttlecock, where they should have hit the shuttlecock to maximize their chances of scoring a point, what kind of stroke type they should use and more. *Radminton* will even be able to provide suggested moves during a live game using Bluetooth earpieces and live streams of camera games.

To confirm the project works at a high level, this will be used along with competitive badminton play. This will also be reviewed by competitive badminton players to ensure that the feedback and suggestions are correct.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

This is the proposed project timeline for our senior design project.

Task Breakdown

1. **Image Tracking**
    1.1. Player Tracking
    1.2. Ball Tracking
    1.3. Court Identification
2. **Real Life Coordinates**
    2.1. Player Location
    2.2. Ball Location
    2.3. Court Location
3. **Object Trajectory**
    3.1. Player Trajectory
4. **Suggestion**
    4.1. Player Location Suggestion
    4.2. Return Location Suggestion
    4.3. Stroke Suggestion
5. **Web App**
    5.1. Website
    5.2. Web Service

Task Dependencies

Task	**Dependencies**	**Estimation**
1.1 Player Tracking	None	3 weeks*
1.2 Ball Tracking	None	6 weeks*
1.3 Court Identification	None	3 weeks*
2.1 Player Location	1.1, 2.3	2 weeks*
2.2 Ball Location	1.2, 2.3	8 weeks
2.3 Court Location	1.3	1 week*

3.1 Ball Trajectory	2.2	2 weeks
4.1 Player Location Suggestion	2.2	3 weeks
4.2 Return Location Suggestion	2.1	2 week*
4.3 Stroke Suggestion	4.1	2 weeks
5.1 Web App	None	8 weeks
5.2 Web Service	2.2, 2.1	7 weeks

Table 4

*This is not a projected estimation, it has been completed already and this was the time it took.

Gantt Chart

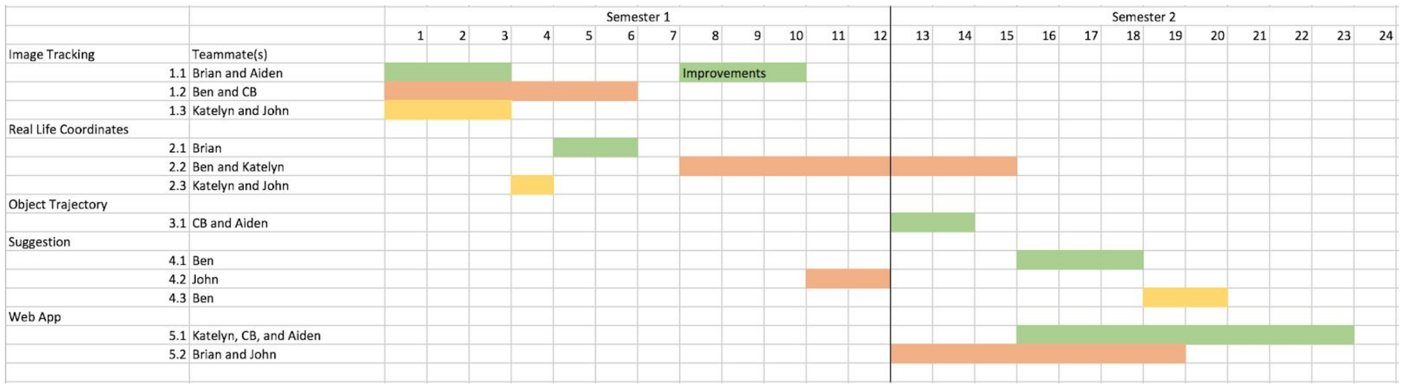


Fig. 1

Above is the Gantt chart showing semester one's completed work and the planned timeline for next semester. Taking into account the exam and vacation weeks, we have about 12 weeks per semester of work. You can see gaps in some people's work in semester one; this is due to personal timelines, working on smaller improvements and maintaining code, and other coursework. Semester two is projected but is subject to change based on unseen difficulties or needed improvements.

The times for semester one were actual work times on deliverables with the specified teammate(s) working on it. Semester two times for the web app were taken from experience on how much time it takes to set up a web service from scratch and hook up and create a website. The other tasks with suggestions or ball locations have known implementation plans so we feel comfortable with our projected estimations.

4.2 Feasibility Assessment

In the beginning, we discussed different outcomes of the project with our client. The client initially talked about wanting to feed existing video into our program to receive feedback. This idea escalated to discussing the feasibility of taking live video and feeding audio instructions to a player on the court. For a while, we wrestled with trying to do it live and how we would need to predict movements without being able to "look ahead" to where the ball was going. This created a stall in our timeline.

While the feasibility of taking live video is not looking very high, the timeline for doing it non-live is good. That removes the complexity of not being able to "look ahead" for the ball location to be able to give feedback to where the player should go and the stroke type. It also removes the complexity of hooking up live video and audio instructions. If the video is not live, then the efficiency of the algorithms is less important which means less time trying to maximise the speed of our programs. In summary, the feasibility of our project being in a place to accept recorded video is high, whereas live is not.

4.3 Personnel Effort Requirements

Effort Point system:

1 - A task with a clear implementation not requiring too much time or resources

2 - A somewhat known implementation or one that requires some research to implement

3 - A somewhat unknown implementation and one that required research and more time

5 - A lot of time to complete or projected to complete, requires a lot of team members to complete in the time frame with much research and deliberation

Task	Effort Points	Explanation
Player Tracking	3	The path for implementation was somewhat straightforward however took a lot of trial and error
Ball Tracking	5	Due to the small size of the birdie, there was a lot of work done to track it and also get a video of high enough quality to track it. It took over a month to complete and a lot of research.
Court Identification	3	As the court was one of the most important objects needed to be identified and the court needed to be found first, development required research into how to identify the court.
Player Location	1	A simple implementation using the court reference to convert to a location, didn't take a lot of resources or time.

Ball Location	5	This required a lot of deliberation amongst the team and if it was feasible to do this live described above in 4.2. Due to the research and discussion that went into it just to decide what the best route was and it is yet to be implemented.
Court Location	2	Required research into how to use the court as a reference frame for the player and ball locations however was quick to implement.
Ball Trajectory	3	An unsure implementation but the investigation time will be capped at a week since it is not a high priority.
Player Location Suggestion	3	A known implementation since Return Location Suggestion we have completed and we can use that as reference. More difficult since ball location is more complicated with timing in the volley.
Return Location Suggestion	2	Required a lot of deliberation and time to figure out what this was dependant upon and also architecture changes to service this functionality.
Stroke Suggestion	1	A somewhat simple implementation that won't require that much time to complete.
Web App	5	This is yet to be discussed in length. It will require conversations on design amongst the team and with the client to do what is easiest but also what users want. It is large and will take multiple months to create the UI and functionality.
Web Service	5	A somewhat unsure implementation and will require a lot of time and resources to build up a web service from scratch.

Table 5

4.4 Other Resource Requirements

Since it is a coding project most materials are just IDEs or libraries. We all used PyCharm as an IDE and used the OpenCV library. All programming and processing were done on our personal computers. In order to record a badminton game, we borrowed equipment from our client to play a game in a gym on campus and a camera on a teammate's phone.

4.5 Financial Requirements

All of the software and hardware we have decided to use is free to download or available for free because a developer already owns it or we know someone willing to provide it. PyCharm is the IDE we use which is free with a commercial edition as we are all Iowa State University Students. All developers own computers to write and run our project on. Cameras can just be phone camera setup using tripods. Therefore no expenses are reported for this project.

5. Testing and Implementation

Testing is an extremely important component of most projects, whether it involves a circuit, a process, or a software library

Although the tooling is usually significantly different, the testing process is typically quite similar regardless of CprE, EE, or SE themed project:

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study for functional and non-functional requirements)

2. Define the individual items to be tested

3. Define, design, and develop the actual test cases

4. Determine the anticipated test results for each test case 5. Perform the actual tests

6. Evaluate the actual test results

7. Make the necessary changes to the product being tested 8. Perform any necessary retesting

9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you've determined.

5.1 Interface Specifications

Our project will require interfacing with a server with a database and web service. The database will be written in SQL. The web service will be written in Java with Spring-Boot; it will communicate with the database using JDBC. We have chosen these technologies because of our familiarity, and the fact that Spring-Boot is an industry-leading web application framework.

5.2 Hardware and software

Our Python code is being tested with the unittest library. It is the most popular unit testing library for Python, and it provides an intuitive, clean system for testing. It also includes mocking tools.

Our backend unit testing will be performed with the JUnit library. Similarly to unittest in Python, JUnit is the most accepted library for testing in Java.

Our backend integration testing will be performed with JUnit and Mockito. JUnit is being used for the same reason as unit testing. Mockito, however, is going to provide us with tools to mock out functionality in the backend so we won't have as much overhead in our integration tests.

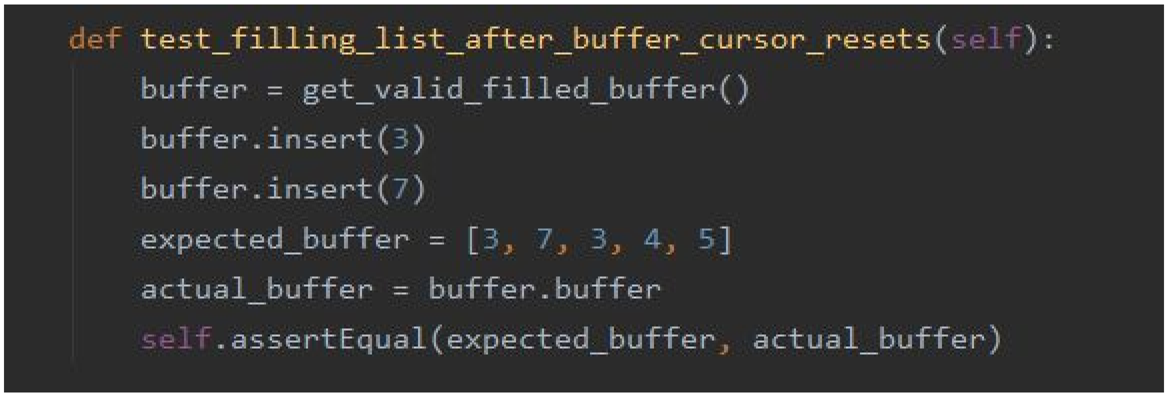
5.3 Functional Testing

Test Plan

Our project will consist of two testing components: unit testing and integration testing. Our unit testing will cover our source classes, covering all non-trivial functionality. It tests each individual component to ensure the small pieces work. Our integration testing will check if the system works in its entirety. We are using the unittest library for Python unit testing, and JUnit for Java unit testing. Integration testing will only be done in the Java backend, so it will be using both JUnit and Mockito.
For Python unit testing, we will write the tests after each component is written. The reasoning behind this approach rather than test-driven-development is that the green state for our code will not come down to exact results. For example, we don't know what bound to expect to capture around each player for our player identification. If we write the code first, we can find a practical tolerance for the location to set for unit tests.
Our Java backend will be taking advantage of test-driven-development. It's going to be more practical to implement since there are exact results we can test for.

Test Samples

At this point, we only have Python implemented. Thus, we do not have samples for the backend. The test shown below is what a typical test case in our Python code looks like. We use highly descriptive names to maintain clarity.



```python
def test_filling_list_after_buffer_cursor_resets(self):
    buffer = get_valid_filled_buffer()
    buffer.insert(3)
    buffer.insert(7)
    expected_buffer = [3, 7, 3, 4, 5]
    actual_buffer = buffer.buffer
    self.assertEqual(expected_buffer, actual_buffer)
```

Fig. 2

5.4 Non-Functional Testing

Performance

- The number of frames we can process per second

Usability

- How user-friendly our design is, and whether or not it meets our client's standards.

5.5 Process

To test performance, we observe the amount of time it takes for each frame. If it is running at an unreasonably slow frame rate -- less than 15 fps -- we will examine the cause. For example, when our player tracking was bottlenecking the application, we redid it because our initial implementation was using a slow, built-in OpenCV function. Other than finding a new implementation, we have also used parallelising and typical optimising as performance-boosting strategies.

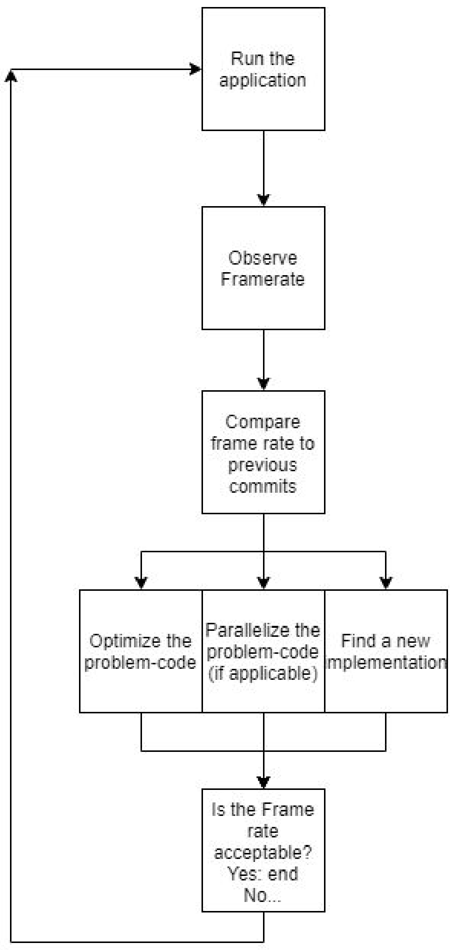

Run the application
Observe Framerate
Compare frame rate to previous commits
Optimize the problem-code
Parallelize the problem-code (if applicable)
Find a new implementation
Is the Frame rate acceptable? Yes: end No...

Fig. 3

Usability testing is based on how easy our application is to use, and if it meets the needs of our client. To do this, we demo the application weekly and get feedback from our client.

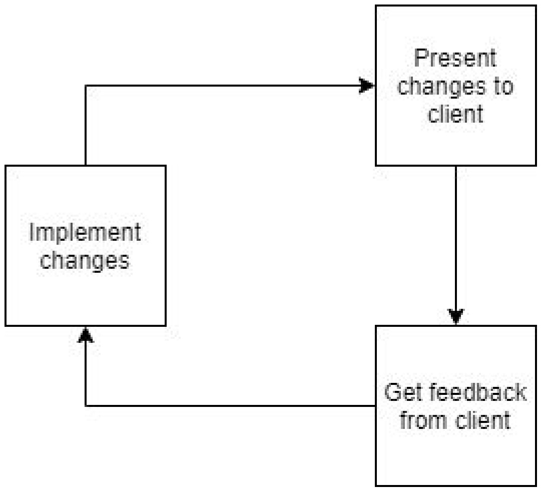


Fig. 4

5.6 Results

Successes

- Player Tracking
  - Player tracking runs without fail. There are no false positives, nor are there ever instances when the players aren't tracked.
- Shuttle Tracking
  - Shuttle tracking has been successful; it runs well enough to get meaningful data.
- Homography
  - We can quickly translate angled coordinates to top-down coordinates.
- Shuttle Return Location Recommendation
  - We can provide meaningful data as to where each player should return the ball
- Court Line Recognition
  - Each line on the court can be identified in a trivial amount of time; it only runs on the first frame of the video.

Failures

- 3D Transformation
  - We have put a lot of time into trying to find a way to transform 2D coordinates into 3D. These efforts have been futile, unfortunately. There is no reliable way to translate 2D coordinates into 3D coordinates. Next semester, we will evaluate the need for 3D at all; we may be able to get the data we need from 2 dimensions. We will also explore getting a second camera, which will enable us to get 3D coordinates.

Learning Outcomes

- Some of our members have never used OpenCV or computer vision in the past, so this project was a good learning experience for computer vision.
- Some of our members haven't used Python, and this project provided a good way to learn it.
- We learned a lot about architecture. Clean architecture was a high priority for us, and that has led us to learn more about architecture patterns.
- We learned about the logistics of coordinate transformations by exploring 3D and 2D transformations.
- Most of us came into this project knowing almost nothing about badminton. We've learned a lot about the sport from this project. We've also gotten together and played it.

Implementation Issues

- Our original architecture was poorly set up. We spent about a week refactoring it and ended up with a design we liked.

6. Closing Material

6.1 Conclusion

In trying to actualize this project, the team has constructed a substantial demonstration that shows our ability to correctly identify player locations, shuttlecock locations, model court dimensions and suggest basic actions to the player. The application has also already been largely architectured, giving us a good frame to work with going forward.

However, our ultimate goal goes far beyond this demo. *Radminton's* goal is to be an easily accessible web application capable of receiving video and giving suggestions to users. Beyond that, we want to provide real-time feedback to players using Bluetooth headphones.

The best plan of action in achieving this goal is to build a powerful identification and suggestion engine that can then be connected with an easy-to-use web interface. As our engine has started to actualize, web interface creation will begin and be the primary concern of a small portion of the screen. This will allow us to deliver a more user-friendly demonstration than anyone with a phone or computer can easily check out and follow.

From there, the power of the application will continue to grow and deliver all the features discussed in this document.

6.2 References

[1] Features2D + Homography to find a known object — OpenCV 2.4.13.7 documentation. (2019). Retrieved 9 December 2019, from
https://docs.opencv.org/2.4/doc/tutorials/features2d/feature_homography/feature_homography.html

[2] Fishified/Tracker3D. (2019). Retrieved 8 December 2019, from
https://github.com/Fishified/Tracker3D/blob/master/DLT.py

[3] OpenCV: Contours : Getting Started. (2019). Retrieved 8 December 2019, from
https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[4] OpenCV: How to Use Background Subtraction Methods. (2019). Retrieved 8 December 2019, from
https://docs.opencv.org/master/d1/dc5/tutorial_background_subtraction.html

[5] Python PEP 8 -- Style Guide for Python Code. (2001). Retrieved 8 December, 2019, from
https://www.python.org/dev/peps/pep-0008/