Radminton

Decision Support in Racket Games

Final Report

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

Engineering 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.
- Users must provide a video they recorded for non-live games.
- 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 for 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.

ReactJS with Redux

- For our web application, we chose to use ReactJS with Redux. Many of us hadn't used ReactJS before so it was something new to learn despite being similar to frameworks like Angular. We also used Redux as a state manager to pass information between components which was a new concept for many.

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, where they should have hit the shuttlecock to maximize their chances of scoring a point, and what kind of stroke type they should use and more.

1.3 Operational Environment

Since *Radminton* is hosted on a web server it can operate anywhere a modern smartphone or laptop has an internet connection. 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*. However, there is one caveat. Objects in the video other than the players, net, birdie, rackets, and lines can cause issues within the court detection. These objects can cause the application to not work appropriately.

1.4 Requirements

Functional

- Videos 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.
- 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.
- 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 been before returning/dropping the ball.

1.5 Intended Users and Uses

*no longer applicable since we didn't have the time and resources to do real time processing

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 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 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 usable 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.
- The maximum number of players on the court is two
    - Typical badminton games can be up to four players. Due to different priorities, we were not able to account for four players.

1.7 End Product 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 is 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 Location Suggestion	An ideal location is suggested for the player.
Return Location Suggestion	An ideal area to hit the ball towards is suggested.
Return 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 much more.
Feedback from Video	Given a video of gameplay, our application can give suggestions on how to improve the user's skill and strategy while playing badminton
Website	The project will feature an easy-to-use website as a means to interact with the software that analyzes a video and returns a suggestion
Web Service	The project needs a way to support the use of the website, delivering all the functionality as a service.

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 their skill and strategy due to our use of modern video processing and analyzation techniques within the Python framework. In both cases, *Radminton* can give back hit by hit breakdowns (formally known as rallies) 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 was delivered on March 8th.
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 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 the 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 has been designed to create a simple way for a user to upload their pre-recorded videos, view past videos, and for each video, view the data associated. This will require a login view so that the session knows which user is accessing it, an upload view, for uploading videos, a view to select which video to view data on, and the data view. The data view has to be capable of showing a processing status for when that video is still being processed. When it has finished the data view shows the video for them to watch, a dropdown including each volley and upon clicking that volley, showing the bird's eye view of the court with the provided player location, ball location, recommended move to location, and recommended hit location for that volley.

2.2 Design Analysis

I will be breaking up the design analysis into tracking, real-world locations, suggestions, and our website.

In regards to the tracking part of this project. Tracking was done with OpenCV in python. The overall accuracy and efficiency of the tracking module was correct. Many aspects of the tracking had to go through multiple iterations in order to find optimal methods. We ended up using background subtraction as our main method of object tracking. This gave us the most flexible and accurate tracking. Overall I think the tracking aspect of this project was a strong suit.

Translations to real-world locations were done using multiple methods. These methods were determined after heavy amounts of research. This reason we did not need multiple iterations of this project element. We had very little issues with our real-world translations.

Suggestions were implemented from the tracking information to give the players feedback. This mostly used the information from the ball recognition and player recognition combined with some logic. This resulted in a lightweight solution to proposing suggestions. Unfortunately, Suggestions were not as fleshed out as we hoped. We originally had hoped to have suggestions for optimal stroke, optimal player location, and optimal return location, but we were only able to implement optimal return location and suggested player location.

Lastly, we have our website. Our website was created with the idea that anyone could visit and upload their videos for feedback. This website was created with React JS and used Django and SQL. Each of these was determined after multiple meetings with our advisor. These ended up being great solutions. Unfortunately, our front end was not as polished as we wanted. Due to a combination of factors, our user interface and appearance is not as complete as we had hoped.

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 Conceptual Sketch

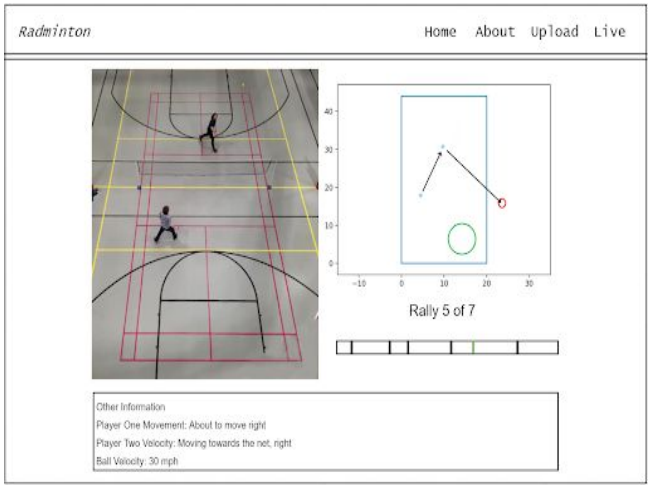

Fig. 1

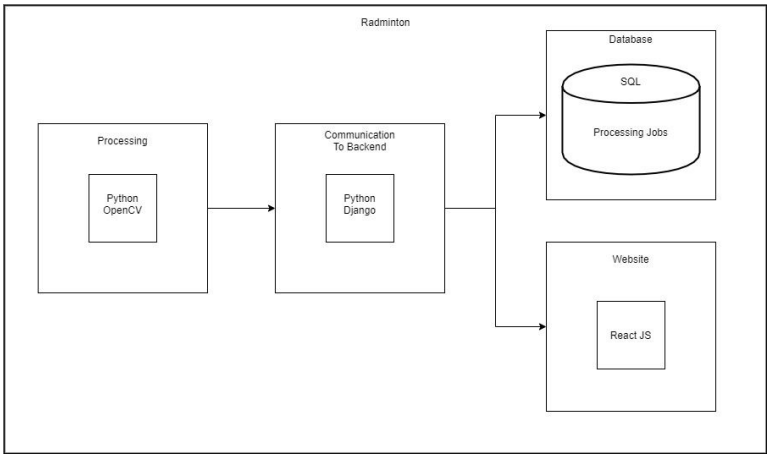

Fig. 2

In order to represent the system two figures, figure 1 and figure 2 have been provided. Figure 1 represents what the intended user interface for a user is. Figure 2 represents how our modules come together. Breaking down figure two you can see that everything starts at the python processing. From this the data is then sent to the backend and website using django. The processing information is stored in the backend while the player feedback, video, and data for the heatmap is sent to the website. This is how the website is able to display the view you see in figure 1. The sketch in figure 1 allows users to skip to specific points in videos where the birdie was hit. The system wide setup as well as the user interface fulfills all requirements we had set and allowed for a good user experience.

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.
For our frontend, no literature was used, only previous work knowledge and advice from our faculty advisor was used to choose dependencies.

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.
For our WebApp we decided upon, with advice from our faculty advisor, on using ReactJS as a base framework. In order to have a base ui for components, we used a components framework called Material-Ui for buttons, etc. In order to manage passing information between components, i.e. a state manager we used Redux which is commonly used with ReactJS. For making ajax calls we decided upon Axios. On our backend, we set it up using django and built up our endpoints.

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 farthest away from the defending player.

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.

Web App - provide a website for users to easily upload and manage their videos for processing

- Website
    - A website that users can login, upload a video, view past processed videos, and for each video, view the volley times and recommended data and bird's eye view of court with the player and ball locations.
- Backend
    - Backend service using our semester one processing logic to create endpoints to receive videos to process and send video data back to frontend

3.4 Possible Risks And Risk Management

Some possible risks of this project are the covid-19 outbreak preventing us from doing anything in person, players being given bad suggestions/feedback. The outbreak has prevented us from meeting in person, which has made coordination much more difficult. However, we managed this by just having virtual meetings in Google Hangouts. Players being given bad suggestions or feedback could result in the user being less satisfied in our project. Since our program analyzes the player's gameplay and provides

feedback any bug or unintended behavior could give poor feedback. This was mitigated by reviewing the suggestions made.

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 and where they should have hit the shuttlecock to maximize their chances of scoring a point. Users can then upload their pre-recorded video to our website and view this advice after the video has finished processing.

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. **Suggestion**
    - 3.1. Player Location Suggestion
    - 3.2. Return Location Suggestion
4. **Web App**
    - 4.1. Website
        - 4.1.1. Setup react app
        - 4.1.2. Upload Video
        - 4.1.3. Loading User Videos
        - 4.1.4. Video Recommended/Volley Data View
        - 4.1.5. Login
        - 4.1.6. Video Player
    - 4.2. Web Service
        - 4.2.1. Upload Video Endpoint
        - 4.2.2. Retrieve All Videos Endpoint
        - 4.2.3. Send Video Recommended/Volley Data Endpoint
        - 4.2.4. Get Video for Video Player
        - 4.2.5. Video Processing Status Endpoint

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 Player Location Suggestion	2.2	3 weeks
3.2 Return Location Suggestion	2.1	2 weeks
4.1.1 Setup react app	None	2 weeks
4.1.2 Upload Video	4.2.1, 4.1.1	3 weeks
4.1.3 Loading User Videos	4.2.2, 4.1.1	2 week
4.1.4 Video Recommended/Volley Data View	4.2.3, 4.2.5, 4.1.1	7 weeks
4.1.5 Login	None, 4.1.1	2 weeks
4.1.6 Video Player	4.2.4, 4.1.1	4 weeks
4.2.1 Upload Video Endpoint	None	2 weeks
4.2.2 Retrieve All Videos Endpoint	None	1 week
4.2.3 Send Video Recommended/Volley Data Endpoint	3.1, 3.2	3 week2
4.2.4 Get Video for Video Player	None	1 week
4.2.5 Video Processing Status Endpoint	4.2.1	1 week

Table 4

Gantt Chart

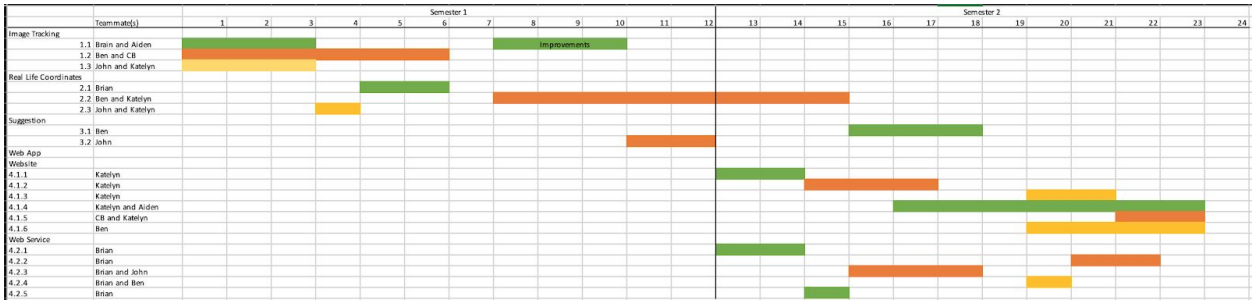

Fig. 3

Above is the Gannt chart showing two semesters of completed work. 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 from unforeseen bug fixes, working on smaller improvements, maintaining code, personal timelines, or difficult coursework that a student had to work on that week. This Gannt chart as opposed to our semester 1 report, includes the web app tasks broken down into projected work. This is due to us having a better sense of the work.

The times proposed were decided upon in planning sessions. Discrepancies between our proposed and actual times of work and deliverable dates were due to a few bugs. One being the introduction of a database meant that the time splice data was being sent incorrectly and not in proper json format, therefore the recommended view was delivered much later than expected. You will also see that John has some gaps in the second semester because he was focusing on maintaining and improving processing for some issues we saw when developing our web app.

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 analyzing live video is not looking very high, the timeline for analyzing non-live video 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 dependent upon and also architecture changes to service this functionality.
Setup react app	3	Setting up the initial frontend app with the frameworks we need. Required research into what frameworks to incorporate to meet our

		needs on the web app.
Upload Video	2	Users can upload a video and send off a processing job on the backend. Will include absorbing a endpoint but not a lot of uncertainty on implementation
Loading User Videos	2	On load and after login, the user will have access to the past videos they have uploaded and see their Will include absorbing a endpoint but not a lot of uncertainty on implementation
Video Recommended/Volley Data View	5	Will require research on frameworks to use to show a court view on the frontend. Uncertainty on how to implement and a lot of work means 5 effort points
Login	1	On page load, the user must input their name and sport in order to view their data and send off videos to processing using their info. Does not require connection to backend since we don't have a password requirement, only saves user and sport as a state variable
Video Player	3	The user can view the video they uploaded and past ones. Also has functionality to skip to the volley they selected to view data on. Difficult to implement and uncertainty on what framework to use so it warrants 3 effort points
Upload Video Endpoint	3	Endpoint to accept video data to begin processing and set off a job request. A lot of work warrants a 3.
Retrieve All Videos Endpoint	5	Requires setting up a database to store a user's past videos, their processing status, and data. Setup time and research warrants 5 effort points
Send Video Recommended/Volley Data Endpoint	3	Will send the data on when volleys begin and within each volley the player and ball location and recommended return and move to location for a specific video. A lot of work warrants a 3.
Get Video for Video Player	3	Requires research into how best to deal with video data. Requires research and some uncertainty on implementation so a 3.
Video Processing Status Endpoint	1	Used to poll for status on the frontend to know when a video had finished or failed processing. Clear implementation and not difficult.

Table 5

4.4 Other Resource Requirements

Since it is a coding project most materials are just IDEs or libraries. We all used PyCharm and Webstorm as an IDE, used the OpenCV library for processing, Django for backend, ReactJS, Axios, MaterialUi and redux as frontend technologies. 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, available for free or already purchased. IDEs we used are 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

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 Python using Django. We decided to go with Django because the main processing part of the application is already written in Python. It's also the best web-app framework that's available in Python.

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. This is the only tool we ended up using for testing.

5.3 Functional Testing

Test Plan

Our project consists of unit testing and acceptance testing. Our unit testing covers basic data structures that, if broken, would have severe detriments to the project. For Python unit testing, we wrote 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. We don't have many automated tests since we had rapidly changing requirements. We decided most of our testing would be done by checking for acceptability by actually using the application. While this isn't ideal, it freed up time to implement more features.

Test Samples

The test shown below is what a typical test case in our Python code looks like. We use highly descriptive names to maintain clarity.

```
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. 4

5.4 Non-Functional Testing

Performance

Our functional testing consists of checking if the performance is acceptable. This includes processing time and the fps of individually running processing jobs. We also look for usability, such as checking if anything breaks under certain conditions in the production application. We also look to see if our design is meeting our client's needs.

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.

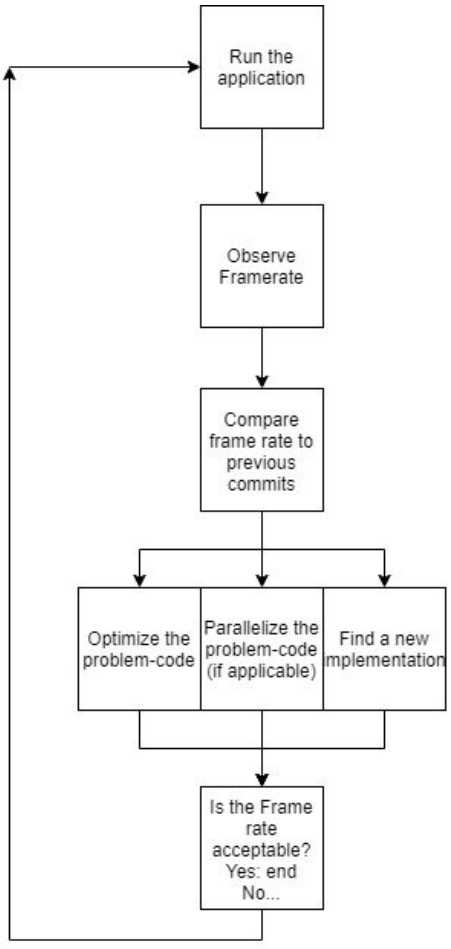

Fig. 5

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.

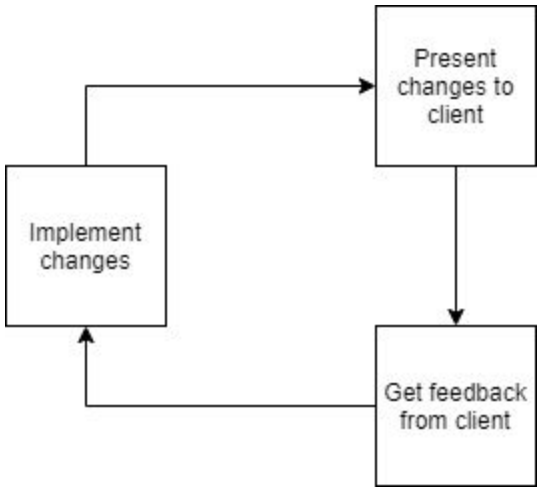

Fig. 6

5.6 Successes and Failures

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.
- Backend
    - The backend can run multiple processing jobs, and queues jobs that would overload it until another one finishes
    - There are several endpoints the front-end uses to allow users to process and analyze videos.

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. Results

6.1 Using Radminton

Using Radminton is a simple process, but there are a few things the user will need to make the most out of the application. Once the user has prepared a video they would like to analyze and has finished reading through the following instructions, they will be able to use Radminton quite easily.

Preparation

Before using Radminton, the user should make sure they have recorded a video of them and another player playing badminton. While Radminton does not have a game length recommendation, the user should be advised that Radminton works best when videos are under 10 MB in size. This means ultra-high resolution cameras, like those found on modern smartphones, may not be the most suitable capturing device, especially if the user plans on capturing a longer game.

The optimal capture angle for video capture would be a top-down view. However, this view is not realistic for most users, so the court detection algorithm is able to handle and adjust to a range of possible video angles, but a 45 degree angle is recommended.

It is important the user plays on a court with red boundary lines and with a preferably bright, neon-green shuttlecock. If the court cannot be identified, a fake court will be drawn on the video to compensate, which may yield inaccurate results. Similarly, a shuttlecock that is more darkly colored may yield inaccurate position data.

Finally, the user must have access to the Iowa State University virtual network. This means the user must be on campus, connected to ISU internet or have access to the network through a VPN connection. This is due to the website being hosted on the ISU network.

With these considerations in mind, the user is ready to use the Radminton website.

1. Sign in

To access the website, the user should open their browser and go to http://coms-402-sd-5.cs.iastate.edu:5000/. From there, they will be greeted with a sign-in page. The sign-in page does not require an account. Simply enter your username and the sport you're interested in getting advice on (at this time, badminton is the only supported sport).

2. Uploading Videos

If the user is new, they will be greeted with a prompt suggesting they upload a video. Otherwise, previously uploaded videos will appear indicating their status on the left side of the screen. Clicking on the upload button will bring up the user's file explorer, where the user can select their video. After selecting the video, it will be uploaded to the website and a link to the video will appear on the left side. The video will say STARTED, meaning analysis is still being done and recommendations are coming. In the meantime, the user can watch their video by clicking on the link. Videos that are completed by the server will update their status to say COMPLETE, and then the user will be able to view the feedback data.

3. Using Feedback

Once the video is done being processed, a dropdown will appear to the right of the selected video with times indicating the start and end of a rally (the shuttlecock being hit back and forth without hitting the ground). If the user clicks on one of these times, they will be presented data such as player positions, shuttlecock positions, and recommended player and shuttlecock positions. Below this, a graphic of the court will show where the player should return the shuttlecock to. A button can also be used that will take the user to that point in the video (Note: This feature does not work on videos more than 10 MB in size).

4. Other Features and Uses

The above steps describe the basic process of uploading videos and getting feedback. Aside from that, users can use Radminton as a place to store their badminton games and feedback. Using the same login

fields as when they uploaded their video, a user can return to the website and find old videos they have uploaded, along with the recommendations made for that video.

6.2 Alternative Versions

Throughout development of Radminton, several design plans were scrapped in favor of what the website is now. This ranges from simpler considerations such as the look of the recommendation view on the website, to more major decisions such as whether a mobile app or a website is more preferable. This section will go over what the team found to be the most valuable to reflect on.

One of the biggest changes in the development of Radminton is the choice to remove machine learning as a component of the project. In its initial phases, the team thought a machine learning component could help in giving recommendations to players by learning from the videos given to it. Ultimately, this piece of the project was scrapped in favor of a more heuristically driven recommendation system. A machine learning component would have required far too much team training and setup time for very little advantage. Heuristic advice, that is advice given based on designer rules rather than computer generated rules, was capable of giving enough satisfactory advice for the project with far less overhead.

This decision, in retrospect, was the right one. However, it may have been worth looking into artificial intelligence as an identification means rather than a suggestion one. This is due to the amount of time it took to get accurate results using a pure computer vision model.

Another important decision was whether to make a website or an app to deliver Radminton to users. The client of the project felt a mobile app was more appropriate, and while this may have been a good option, it was decided a web application should be made instead due to the team's lack of mobile experience. A mobile application would have been a bigger undertaking as compared to a website, which the team had far more experience in.

Finally, the team had to realize not everything originally planned was going to make it into the project. This included live audio feedback, 3D positions, and direct video to upload. The team did not have the time to add such substantial features. This can be attributed to the unexpected amount of time needed to get precise readings from the badminton videos. Many different approaches to ball positions had to be used and even simple features of the video like court lines became more complex than originally expected due to the number of lines a typical gym court has.

3D positions could not be added because two cameras would have been necessary to achieve the feature. The team decided early on that this severely cut into the user experience of being able to get analysis on a video, since the user would need two cameras to use the website. It's hard to say whether this was a good decision or not, and there are arguments for both. The rationale behind choosing a single camera setup was sound, since the project was more user focused. If the team had decided to go in a more specialized and accurate direction, two cameras would have been a worthy undertaking, and may have resulted in the core service being higher quality. This project proved how important big questions like these are.

Direct video to upload capture was not completed again due to time constraints. This feature would have required an investment in learning mobile technology, which would have been a drain to our worker

resources. This drain would have been too costly since brain power was needed for several large areas of the project including the website, web service and core identification and suggestion technology.

6.3 Learning Outcomes

Through a year of collaboration, design and development, the team has learned a lot. Three major areas include architecture, full stack development and deployment, and management. Unlike other classes, it wasn't algorithms or techniques that made the biggest impact, but rather how the project was organized, work was distributed and code was maintained over a year's time.

For much of the team, this was the first time working on a large scale project from scratch. This meant code standards and writing with a design in mind were imperative to the health of the project. Some pieces of code were easily understandable from the beginning, while others had to be peer reviewed and modified to follow the guidelines the team set for itself. The value of doing this cannot be understated; as more and more features were added, the complexity grew at a rapid pace. Luckily, the team was focused on architecture from the beginning, and the team worked together to make sure we were all writing code in line with the team's expectations. Much of the team became much better programmers thanks to this project, since so much of the team's academic career has been spent writing programs in relative isolation.

The second learning experience was the struggle of building a full stack application from the ground up. Having three large components (a frontend, a web service, and a suggestion/identification engine) made the project quite complex with many moving, intertwined parts. Thanks to the architecture mentioned earlier, the team was able to keep this manageable, but there was still a lot of learning in the process of Radminton construction. Team members had to learn how to set up their workstations for development, how to stay up-to-date with the latest code versions, manage multiple programming languages and frameworks, understand the project's lifecycle and more. The team learned how to make sure everyone was aware of the newest changes and features to ensure no one was left behind.

Following that, was management. The most illuminating part of the project was the impact of proper management of the team and the members within it. Given that we are all of the same graduation status and have relatively similar experiences in development, leadership and learning how to respectfully criticize and praise each other was something everyone had to become comfortable with. Oftentimes established time targets were missed and it was up to the team to work with each other to decide how to move forward in those situations. The success of the project depended on everyone in the group putting in their best effort, and team members had to learn how to not just do that themselves but promote that attitude within the team.

Furthemore, the team had to be responsible for making it to advisor and client meetings on time, reporting their statuses truthfully, and asking questions about any pain points they had encountered. Being open about failures is not easy, but this project taught us the value doing so. The team's adviser and client was tremendously helpful and understanding in these situations, which taught the team the value of owning failures and learning from them to bring about future successes.

Overall, the team has become much better at writing software thanks to the lessons mentioned above.Future projects will be executed much better thanks to the knowledge gained on software architecture, full stack development, and team management.

7. Closing Material

7.1 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.

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.

7.2 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/