Player Detection and Tracking

Group 6
Spring 2021

Player Detection and Tracking is a project by Diana Bisbe, Brendan Boissineau, Itzik Efraim, Joshua Loufek, Mark Trinidad and sponsored by Dr. Khurram Soomro.

Table of Contents

0 Prelude for Future Senior Design Groups

Originally this project contained within it's scope both the video annotation portion and the footage query portion. However, as we started to work into the second semester it became clear that the scope of the project exceeded our capabilities within the time limit. It was instead decided that we would continue to work on and refine the annotation portion to a higher quality rather than splitting our workload and developing a subpar product.

There are still some elements of this document that reference how we had planned to have both sides of the application communicate and what each of their roles would be. It has been decided to have that documentation left in so that a future senior design group can easily reference it and understand how the side of the application they will be working on relates to the work that has already been done.

Sections involving planning directly relating to the future senior design group will be marked with a (FUTURE) label.

We hope that this document serves you well as a starting point in your journey through Senior Design.

1 Executive Summary

Many of us watch and enjoy sports and understand the insane time commitment that professional athletes put forward to never stop getting better every single day. However, the athletes playing on screen are only half the story. For sports coaches training their team is so much more than just honing the skills of their athletes. An often underrated and looked over part of the training process is the tedious but incredibly necessary process of gathering and synthesizing information.

The analysis and understanding of information gathered from past games is a highly valuable and perhaps even necessary part of a professional athlete's or team of professional athlete's training. The information gathered can be used in a variety of ways, from learning from and correcting one's own past mistakes to studying an upcoming opponent and attempting to identify their strengths and weaknesses. The useful potential is truly immense.

However, despite its usefulness and importance, the gathering and analysis of relevant sports data is not an easy process and can be quite tedious to perform. In terms of soccer, which will be the main sport of focus for this paper, generally speaking each team plays almost 40 games throughout the course of a regular season wherein each game lasts 90 minutes or perhaps even longer if it goes into overtime. Now take into account the dozens upon dozens of teams that play each season whose gameplay you might want to analyze and you are confronted with a massive pile of data you'll need to sift through where only a small part is relevant for what someone will be searching for, and all that's still for only a single season. There has to be a better way.

Player Detection and Tracking is a website application that aims to serve as a tool for two primary purposes: 1. gameplay footage analysis and classification to be stored in a database for simple and convenient use in the future, and 2. a user-friendly database query tool for finding gameplay clips relevant to the user's search. The crux of the application is the first part, wherein the user tasked with gathering data from a game is aided by various object tracking algorithms which work mostly automatically with as minimal user input as currently possible.

Once a game has been analyzed and the data safely stored it then becomes easy to get information out of it in the future. Since it's all stored in a database a simple query through any of a variety of methods would instantly return all relevant footage based on what the user is searching for. This would dramatically cut down on time spent on the data gathering front in the future.

2 Project Narrative Description

2.1 Narrative Description and Function

An important component of competitive sports is reviewing past broadcasts both of your team and of that of your future opponents. Looking back can give you some of the most valuable information on how to proceed and prepare for future games. However, sports broadcast footage can be limiting and analyzing it by hand can lead to significant levels of human error, especially when it comes to missing or misunderstanding an event on screen. But most importantly, sports broadcast analysis by hand is incredibly tedious and time consuming. As they say, “knowing is half the battle”, so a convenient way for sports teams and analysts to be able to gather data from past broadcasts more efficiently and with a greater level of accuracy would be invaluable.

We want to solve this issue by creating an interactive web-based platform where soccer and football broadcast footage can be uploaded and more easily dissected and analyzed. The platform will be split into two major sections: the video analysis dashboard and the database footage query. Of note, the database query portion is not within this project’s scope. However some of it is still detailed here for a future group that will complete that part.

First and foremost is the video analysis dashboard. This is an inwardly facing part of the application that deals with the tracking algorithms and manual review and is only accessible to members of Dr. Khurram Soomro’s Sports Science AI team. The dashboard is backed by automatic machine learning algorithms trained to identify and track players on the field and translate that tracking information into data that can be manipulated and utilized in a variety of ways. The simplest of information gleaned (but no less valuable) is player position on the broadcast video which can be translated into a 2d field view to see each and every player’s exact position on the field at any time.

An essential component of the dashboard is the manual review function. After a broadcast video has been uploaded and analyzed by the automatic detection and tracking algorithms it is then available for manual review. While detection and tracking algorithms can ease the workload burden from someone doing it all by hand, they aren’t foolproof because their accuracy can waver and fail. In addition, our algorithms do not possess the inherent ability to recognize the player they are tracking, they only know where a player is, but not which player they are tracking. Due to this our application will allow the user to manually review the tracking footage to search for errors and correct them and assign detection tracks to the player it is tracking. For instance, a player

tracker could lose sight of a player for some length of time, and when they are redetected the algorithm will start a new track despite it being the same player. A user would be able to connect both tracks to the player to enforce consistency between the two and make sure the algorithm recognizes them both as the same player. In addition the user is able to create a new track to fill in the gap of time where there was a gap in the detection and assign that track to the player as well to fill in gaps in the automatic tracking.

With all of these annotation and correction tools available to the user it then becomes possible to completely and accurately track and log all relevant player positional information throughout the entire course of the game.

Secondly we have the database footage query. This is the outwardly facing part of the application that any normal, authorized user would be able to access. Here the user is able to query the database in a variety of ways to quickly and easily pull relevant footage based on their search. For example, a user would be able to specify one or more teams they wish to find clips for while also specifying where on the field the camera should be looking for those clips, or which players should be in the scene.

2.2 Individual Statements of Motivation

Growing up I spent most of my time playing soccer. When I was 13 years old, I left my parents' house and moved to a soccer academy to pursue the dream of becoming a professional soccer player. At the same time, my interest in computers was growing and when I started to think about what I want to study in college, the decision to go with computer science was a no brainer. Up until now, these two fields that are huge in my life, co-existed in parallel and never had anything in common. When I watched the presentation about this project, I knew immediately that this is the project I want to do. Not only it involves my biggest passions, but there is also a computer vision component to it – an area that I am planning to pursue a master's degree in.

-Itzik Efraim

I loved playing soccer throughout most of my elementary school years and have also played a variety of other sports since then, including both lacrosse and tennis. Also my parents are from Pittsburgh, Pennsylvania and as such I grew up in a Pittsburgh Steelers household watching Sunday night football every week it was on. Since then, while studying computer science I've taken a large interest in both machine learning and robot vision and have really enjoyed learning about and studying the application and implementation of these fields of study and how they can even combine to work together. As such, this project perfectly blends two of my interests and gives me a fantastic opportunity. I will not only be able to learn more conceptually, but also get some more hands on experience working with and combining these algorithms into a polished and final project.

- Joshua Loufek

I have been a big sports fan growing up, whether it is football, basketball, or even combat sports; I expose myself to sports entertainment on a near daily basis. One thing every sport has in common is the data and analytics behind everything. With the exponentially growing world, the advancement of all data is growing with it. This project caught my eye, not only because of the sports aspect, but because of how interesting, complex, and challenging it seems. While keeping up with the sports world, I have been studying computer science with a minor in math. I have done many projects, including a web-based application, and taken classes that gave me

experience, like Artificial Intelligence. As time went on, I realized that I would love to connect my knowledge to what I enjoy, and this project is a great opportunity for me. I am excited for the challenge and cannot wait to learn and grow from this opportunity.

- Mark Trinidad

One of my biggest motivations was the theme of the project: Applying Computer Vision and Machine Learning algorithms to sports. Since I took my Robot Vision class, 2 years ago, I have thought that computer vision is one of the most impressive kinds of AI. I would like to continue to delve into this field, as I would like to pursue a Master's degree in Computer Vision, or AI, with a track specialization. I am very grateful for the opportunity I have to improve my skill set and work to solve a realistic problem, while I learn more about this rapid changing technology. I have always been a fan of sports, especially soccer, volleyball and baseball. This project could impact the way we get our data from different matches and help improve player skills, or team dynamics.

- Diana Diaz Bisbe

Since first learning about programming when I was 10, I've always been interested in the vast array of different ways you can approach solving problems and creating solutions. Back then, I was interested in making cheats for FPS games and creating random things in Visual Basic, but now that I've pursued Computer Science and have gotten to my senior year, I really want to delve into the depths of Software Engineering. This project has a multitude of different areas that are all extremely interesting and will enhance my knowledge in every aspect. With all of the deep learning and every other aspect that this project is about, I will be able to not only be a part of this project that I will be able to happily preach in interviews and on my resume, but also be able to utilize these skills that I will learn and apply them towards other projects that I become interested in.

- Brendan Boissineau

2.3 Goals and Objectives

There are three main goals on the project:

- Player Detection and tracking:
    - Detect players using Deep Learning Object detection methods.
    - Tracking players on the field by associating player position between frames.
    - Enable real-time tracking.
    - Identify player position on the field and map on 2D field image.
- The web portal which will have two kinds of users:
    - The outside users. (coach, managers, technical staff)
        - Upload sports videos for processing.
        - Search the already-processed video database.
    - Inside users. (Sports Science team)
        - Edit video to trim unnecessary footage
        - Add/Delete new or unnecessary tracks with the online tracking algorithm.
        - Annotate and edit videos after the preprocessing is done.
- Web-portal Search Engine:
    - Search historic games based on player position on the field.
    - Find similar plays given a play video.

The Computer Visions algorithms will be used to obtain from the video tactical information that can be searched and visualized. For example, team formation or player's statistics like average speed, distance covered and number of passes.

2.4 Constraints

- Product must be processed and delivered through the cloud using Amazon Web Services.
- Currently only able to process soccer games.
- Processing does not automatically start. Must go through the Sports Science Team for confirmation.
- Results are not immediate, and processing can take hours to days.
- Project must be completed before the senior design showcase at the end of the Summer 2021 semester.

2.5 Broader Impacts

This project will impact the way training and film study is conducted and team dynamics in sports. It will multiply the possibilities of evaluating the performance of a team, being able to make better decisions regarding players(in case of a coach) or their actual performance (in case of a player) in the training process. It will mean a different way to export the information to share with the technical staff or players.

Users, either players or coaches will be able to check for changes in performance by analyzing the player statistics created out of the information obtained from the video process, which will lead to smarter training to track progress. This statistics could also help the coaches analyze the competitor strengths and weaknesses, and create strategies based on that.

The search engine will allow for coaches or technical teams to provide almost immediate feedback, since they will be able to quarry the data to find a certain moment or movement from the video.

All in all, upon successful completion of the project, this application will help coaches, players, and sports fans easily search, find, and view the many different plays from a refined statistical viewpoint

2.6 Legal, Ethical and Privacy Issues

2.6.1 Legal Issues

When searching through patents related to our project we were able to identify many that relate to what we are doing. The following list includes some of the most relevant patents, however this list is not exhaustive:

- US20080192116A1: Real-Time Objects Tracking and Motion Capture in Sports Events
    - https://patents.google.com/patent/US20080192116A1/en
    - Claims a system for real-time object localization, tracking and personal identification of players in a sports event.
    - Claims a system for automatic objects tracking and motion capture in a sports event.
    - Claims a method for real-time motion capture of multiple moving objects in a sports event taking place on a playing field.
- US5363297A: Automated camera-based tracking system for sports contests
    - Claims an automated tracking system for obtaining a detailed data base, representing the position in time of the players in a sports contest conducted in a playing area.
- US9036864B2: Ball trajectory and bounce position detection
    - Claims a method and system of determining a ball bounce position on a playing surface.
- US8705799B2: Tracking an object with multiple asynchronous cameras
    - Claims a method and system for determining a position of a moving object through use of multiple cameras.

The most concerning patent listed above is US20080192116A1. All three of the claims we listed from the patent do relate to our project, however the use for the data gathered and some of the methodology for gathering the data has significant differences. This patent details that real time object tracking data is to be used for creating and displaying a Virtual Flight Clip from a high resolution recorder. The Virtual Flight Clip is a three-dimensional display of the real sporting event using computer games graphics.

Our method focuses on using already recorded gameplay footage from a broadcast view. We do this through a web-based application and the purpose of gathering data and tracking players is for an entirely different purpose. Our application aims to synthesize and visualize that data for the purpose of gathering individual player and team statistics.

2.6.2 Ethical Issues

The services our application provides are simply a data gathering extension to events and broadcast footage that already exists and is analyzed in different formats and contexts. Nothing that our application allows users to do is invasive or predatory for the users or the people in the footage. In addition, it is not feasible that users of our application could use it to infringe upon the rights of others or use it to learn how to infringe upon the rights of others.

2.6.3 Privacy Issues

The data gathered from our application only pertains to players on the field who are not gaining any additional or unnecessary exposure due to our application. The broadcasts we are using, be them live or recorded, are already in the public eye in the most extreme possible way and we our application will not be heightening that in any way. Our application's primary purpose is to aid in gathering data about players and sports matches in general. The data gathered is not intended to be used against the people or players involved in any capacity.

In addition to the players on the field there are also people on the sidelines and bystanders in the crowd that can be present in the broadcast footage. However, our application is designed to only gather data from players on the field and the machine learning algorithms even filter out non-player entities. This means that the privacy of bystanders are also not being infringed by our application.

3 List of Specifications

3.1 User access (FUTURE)

3.1.1 Requirements

- Allow users to create an account, log in, log out. Information collected should be first and last name, email address, username, and type of user.
- Users can have one of two models: Sports Science Team and Client.
    - Sports Science Team: will have access to unedited videos and can edit out unnecessary footage from a submitted video and confirm the video for processing.
    - Clients: will be allowed to access a query of edited and unedited videos, while also being able to upload to add to the database of videos
- All users can search through the database of all games that have been uploaded and select to go to the related game.
- Clients will have a game view that will give all the finished analytics post process.
- Sports Science Team will have two additional views:
    - A preprocessing view to crop the client's video.
    - A postprocessing view to refine the annotations and provide id to the annotations.

3.1.2 Stretch Goals

- Give users choices and different permissions at admin(coaches) or members(players)

3.2 Video Processing Dashboard

3.2.1 Requirements

The dashboard is split into multiple separate phases with different functions.

- Video Editing Mode
    - This part is used to trim all the excess footage and keep it down to just the core game footage.
    - Users should be able to play the video at normal speed, pause, fast forward, slow down, and be able to go backwards and forwards from frame to frame while paused.
    - Users should be able to remove specific frames and specific sections of frames from the overall video.
    - Also verify basic information uploaded by the user.
- Automatically Generate Tracks
    - After video editing is over this is run once to generate the majority of tracks that will be used in the next phase.
    - Connected to the detection and tracking algorithms to take the video input from the previous step and generate tracks.
    - Each track is generated with a unique ID and separated from each other for more efficient processing in the next step.
- Track Modification
    - When viewing the footage and tracks in the dashboard the video should be in the center with a toggleable overlay that shows the bounding boxes
    - To select a track to interact with the user should be able to:
        - Click on a bounding box on the video overlay on screen.
        - Click on the track in the sidebar.
    - When interacting with tracks you should be able to do the following:
        - Go to the start of the track and view it from start to finish.
        - Delete the track.
        - Edit the tracking contents by cutting out sections.

        - Add a new track.
            - Should be able to choose a start frame to draw the bounding box on and then an end frame where the model will stop running.
        - Assign the selected track to another track.
        - Select multiple tracks simultaneously to assign to another other track.
- Semantic Segmentation and Camera Parameter Estimation
    - Once all tracks are assigned properly, the user then runs the semantic segmentation on the video and tracks to generate the homography and field view.

3.2.2 Stretch Goals

- Video Editing Mode
    - Add editor scroll bar.
- Automatically Generate Tracks
- Track Modification
    - When interacting with tracks you should be able to do the following:
        - Toggle whether the track is viewable.
        - Rename and add notes.
        - Display error if frames overlap when tracks are attempted to be assigned to one another.
            - Have an error view where both tracks are shown simultaneously and choose which one you'd want to modify to resolve the error.

3.3 Client Search Engine (FUTURE)

3.3.1 Requirements

- Upload video for processing
    - User needs to include basic game information:
        - Teams playing
        - Date of game
        - Players
- Basic search engine that can search through entire database and refine by the following properties:
    - Choose whether looking for entire games or want video clips.
    - Specific game ID(s)
    - Timeframe
    - Which team(s). For multiple listed teams two methods of search:
        - Method 1: Games that include any listed team.
        - Method 2: Games where both teams are listed teams.
    - Players involved
    - Advanced options:
        - Player proximity for interactions
        - Location of field view

3.3.2 Stretch Goals

- Refined search engine that you can query by specific players, player locations, etc.

3.4 Back-End API

3.4.1 Requirements

- Amazon Web Services
- Video analysis on the cloud
- noSQL database
    - Store User data
    - Store Game data
        - Player Tracks
        - Potential Video
    - Store labels
- Run machine learning algorithms
- Search engine

3.5 Object Detection

3.5.1 Requirements

- Be able to determine which pixels indicate players/objects and place boxes around them.
    - Given the video stream, use a machine learning library in order to classify the objects on the field.
    - Be able to determine which pixels indicate objects in the video; I.e. a soccer ball.
    - Be able to change label data in order to make corrections/customize.
- Utilizing Tensorflow as the preferred library of choice.
- Determine player features (Jersey number, etc...)
- Be able to filter out unwanted detected objects.

3.5.2 Stretch Goals

- Expand the capabilities to other sports.

3.6 Player Tracking

3.6.1 Requirements

- Be able to track player boxes and their movements given the labeled and boxed video.
    - Be able to track multiple players at once.
    - Be able to change label data in order to make corrections/customize.
    - Be able to specify which player to track.
    - Determine statistics based on tracking (ball speeds, counts, etc...)
    - Be able to detect the two teams and separate their detections
    - Be able to give tracker identification for each bounding box
- If there is no suitable model to use for player tracking, either a derivative or new code may need to be used.
- Utilizing Tensorflow as the preferred library of choice.
- Provide a user friendly web application that allows users to upload videos, view game clips, and edit player positions.

3.6.2 Stretch Goals

- Expand the capabilities to other sports.
- Be able to track the ball.

3.7 Semantic Segmentation

3.7.1 Requirements

- Create a neural network that is trained on field lines with high accuracy
- Use semantic segmentation to label every pixel of frame from the video
    - Determine which pixels are field lines
    - Output that can be used for error based learning along with the homography
- Utilizing Tensorflow or pyTorch
- Choosing a semantic segmentation model such as DeepLab or UNET
- Obtaining a dataset of segmented soccer field images
- Creating a training dataset by manually labeling a large set of soccer images
- Use data augmentation to artificially increase the sample size for increased accuracy

3.7.2 Stretch

- Expand the capabilities to other sports.

3.8 Camera Parameter Estimation

3.8.1 Requirements

- Use the information gained from semantic segmentation to map the frame of the broadcast video onto a two dimensional plane.
- Be able to create a map of the two dimensional top-down view based on the homography determined
- Take a list of coordinates, and width/height of a box from both tracking algorithms and determine the feet placement of a player within that box
- Use the estimated homography to find out the position of those coordinates in a warped view that represents the field
- Utilize the map to determine a player(s) coordinates on the field from the frame
- Output the feet coordinates of every player that is being tracked
- Create a heatmap
- Analyze team formations
- Use a model such as SCCvD or Sportsfield Registration for initial usage
- Analyze model and create our own version
- Save homographies to disk for immediate use whenever required
- Save coordinates of players

3.8.2 Stretch Goals

- Use the boundaries of the field to determine what is a player and what is an outlier
- Expand the capabilities to other sports.

3.9 Online Object Tracking

3.9.1 Requirements

- Be able to use only a few previous frames of information in order to track players in real time.
    - Be able to change label data in order to make corrections/customize.
    - Be able to specify which player to track.
    - Be able to track multiple players at once.
    - Be able to track the ball.
    - Determine statistics based on tracking (ball speeds, counts, etc...)

3.9.2 Stretch Goals

- Expand the capabilities to other sports.

4 Idea Listings

The following section includes short descriptions of creative concepts and ideas from each of our group members about directions to explore and potential solutions to various aspects of our project.

4.1 Diana Bisbe

- After the video is uploaded, display the detected player by placing bounding boxes around the player. The boxes would be different colors for each of the teams: the color of their jersey. We can use openCV for this since it will let us identify specific color masks.

- Display the tracking by having the two videos side by side. (the original one with the detected player and the 2D view of the field generated by mapping their positions. Have the players be represented by dots in the 2D field view, and their colors would be the same as the bounding boxes.)

- Use the getPerspectiveTransfotm function from OpenCV in order to create the 2D field view, which will be an elevated view.

- Another idea to implement is to classify player detections in one of five classes: Team A, Team B, Referee, Goalkeeper A, and Goalkeeper B, using the colors inside the detection bounding boxes.

- Before we start training we need to prepare the data with which we will train our program. For this we will need to have a lot of images, at least I would recommend having 200 images for each of the objects that we want to detect (if in an image we have 3 of the objects that we want to learn to detect, this could count as three images). In our case we are very likely to have more than one player per frame or image to be labeled.

- Using **labelImg** to label data since it has a very user friendly interface. Basically what we do with this program is to open an image, select a box to mark the

object that we want our program to learn how to detect and save an XML with coordinate information.

- To set up a training environment for YOLO there are several tools that can be used to create annotations of the images that will be part of the training set. That is, manually indicate the "boxes" that contain each of the objects, and indicate which of the classes it belongs to.
- If we are training for the YOLO implementation with darknet we can use YOLO Mark for labeling data since it saves annotations in the format expected by Darknet.
- Using Amazon CloudFront to play the media files stored in S3 when the user tries to play a video retrieved from the already processed video database.

4.2 Brendan Boissineau

- Utilizing DeepLab for the atrous convolutions and spatial pyrimidal pooling along with the better information loss and computational/memory usage.
- Using STFCN for the semantic video CNNS through representation warping.
- Nanonets for semantic segmentation.
- U-Net. Better than Nanonets and has improved information loss and keeps computation low.
- Using Tensorflow vs pyTorch inorder to utilize the premade models for DeepLab and others.
- Using Supervise.ly or SuperAnnotate for image labeling if custom datasets are required for training models
- Data augmentation on datasets to artificially increase the sample size so that we are able to have more data for training semantic segmentation models

4.3 Itzik Efraim

- Use pre-trained models for player tracking. In order to save time and computation power, instead of getting our own data and training the model for a long time, we can use a model that is already trained, not necessarily with soccer data, but maybe with pedestrian data or other datasets.
- Manually creating a database and training our own model for player tracking. As opposed to the previous idea, we can train our model using relevant data from soccer matches. However, there are not a lot of soccer datasets available, so we would probably need to create our own dataset, something that could take months.
- Use Tensorflow. We chose to use the Python programming language for a reason. In addition to it being a very user friendly language, it also has multiple useful, well documented, and supported libraries for computer vision and machine learning. One of the biggest ones is Tensorflow, which we can use in our model.
- Use OpenCV. As stated in the bullet before, Python programming language is full of useful big libraries. Like Tensorflow, OpenCV is another great option we can use when thinking about creating a computer vision model.
- Implement a tracking algorithm that has its own detection model.
- Implement a tracking algorithm that is dynamic when it comes to the detection model it uses
- If we decide to train the model with our own data, we can either label the data ourselves or use a service like AWS to pay and have them label our data.
- Implement machine learning models using AWS SageMaker
- Implement machine learning models using AWS DeepLense

4.4 Joshua Loufek

- Website having the ability to form “teams” with one administrative user/manager so that multiple users can join the same team and have access to all the same recordings and manual footage modifications.
- Have the website be able to accept videos from a variety of sources: direct uploads, google drive, youtube link, etc.
- Be able to use livestream footage for online tracking.
- Allow the user to download the modified footage, graphs and any other type of created media for use outside the application.

4.5 Mark Trinidad

- After the post processing, manually place a tracking box on a player and automatically connect his track.
- During the processing, be able to automatically categorize the players based on jersey color.
- During preprocessing, automatically trim footage that is not necessary, either through viewpoint not being field or time is not running.

5 Division of Labor

5.1 Diana Bisbe

- Object Detection Algorithm Implementation
- Front-end Video Editing

5.2 Brendan Boissineau

- Semantic Segmentation
- Camera Parameter Estimation

5.3 Itzik Efraim

- Multiple Object Tracking Algorithm Implementation
- Research and Integration of Amazon Web Services.

5.4 Joshua Loufek

- Project Manager
- Real-Time Object Tracking Algorithm Implementation
- Database Design and Implementation

5.5 Mark Trinidad

- Design the web application as well as research technologies for it.
- Research and Integration of Amazon Web Services.

6 Research

This section details the research done for each major aspect of our project. This includes information on which language was chosen for each part, which algorithms we considered for each detection and tracking challenge, a summary of relevant information to help inform the reader as to how each part functions, and ultimately which algorithm or solution was decided upon to handle each section.

6.1 Machine Learning Languages and Libraries

Much of this section was a forgone conclusion before we had really even started our research. The power and capabilities of Python as a language cannot be understated, especially as it relates to machine learning. Python is an easy to use, read and comprehend language for anyone familiar with coding. Plus, the modularity and flexibility provided by the massive variety of packages and libraries you can import is incredibly useful.

Python has access to two incredibly useful deep learning libraries in the form of tensorflow and pytorch and two machine learning libraries in the form of pandas and scikit. Another handy library is opencv which is specifically useful for our project because it allows the user to very simply and easily work with images.

6.2 Detection Tracks

6.2.1 What is a Detection Track?

First, it's important to understand the difference between object detection and object tracking. Object detection is the capability for an algorithm to detect the existence and location of a specified object in a frame or video. It doesn't necessarily remember it's detections from frame to frame, only if an object (or objects) exist in the current frame and where they are. Meanwhile, object tracking remembers the location of each detection from its previous frame and attempts to use that saved information and the slight difference between the two frames to have the detection(s) follow the object as it moves and record the locations.

A detection track at its core is the set of detection locational data that's supposed to be related to one specific object. The information stored in the detection track includes the location on the screen and the frame that it appeared on. Individual detection tracks don't don't necessarily cover the entirety of an object's appearance on screen and they can also have gaps in them where the frame doesn't have any information about the object's location because it isn't within the current camera's view. Individual detections in a track are sorted by frame wherein the earlier frames appear higher in the list and later frames appear lower in the list.

6.2.2 Automatically Created Tracks

Automatically generating detection tracks is the duty of the player tracking algorithm with assistance from the player detection algorithm. Once the video is edited the aforementioned algorithms are run and these tracks are generated throughout the runtime. There can be multiple players being detected each frame and thus multiple players that are being tracked at the same time, however, the tracking algorithm can handle that, resulting in the simultaneous creation of multiple tracks whenever two or more players are visible. Each of these tracks run with no gaps from a start frame to an end frame tracking one object or player the entire time. Once a player can no longer be detected (be it because they are no longer on screen or the algorithm failed to register them as present in the frame) then that detection track concludes. Once the player is detected again a brand new and unique detection track is initialized to track that player. However, as far as the algorithm is concerned this new detection track has no direct relation to the old detection track that used to follow the player and thus the player returning to the frame might as well be an entirely different object.

6.2.3 Deleting, Modifying and Merging Tracks

Without any form of player recognition to be the foundation for some level of object permanence we are left with many, many individual tracks that each on their own are relatively useless. Over the course of an entire 90 minute soccer game it's entirely reasonable that thousands of tracks, if not more, could be created. Collectively these represent every detection and track automatically generated throughout the entire course of the game, but individually they aren't very useful. Therefore, we have to start processing them to get them into a usable state. This involves doing three primary things:

1. Deleting Tracks
2. Modifying Tracks
3. Merging Tracks

First, let's look at deleting tracks. This is a solution for tracks with catastrophic errors such as glitches wherein the tracker is spazzing out and producing wild and inaccurate results, even if it's still sticking close to a player it's trying to follow. Another prime example includes a detection track that's not actually detecting a player, but some other entity in the camera's view. For instance, referees look very similar to how the players do but we don't want to detect or track them.

Second, let's look at modifying tracks. When examining a track the user might notice a less severe error that doesn't necessitate throwing away the entire track. In a situation like this the user could instead choose to merely trim the track to cut out faulty parts while still preserving the good parts on a frame by frame basis.

Finally, we come to merging tracks. This is how we take multiple tracks which are useless on their own and make them into something we can actually use. At the end of the overall processing segment all tracks should be combined and consolidated into a single track which represents the entirety of all detections and trackings of that player, aka a player track. When the two tracks are merged the individual detections of each track are automatically sorted by frame. This means that it doesn't matter the order in which you merge tracks, because as each new track is merged they're already in the correct order. In addition, I mentioned earlier that tracks can have frame gaps in them where no detections were found. While individual, unmodified tracks generally won't have any frame gaps in them, modified and especially combined tracks are bound to have many gaps in them. This isn't a bad thing, it's simply a result of how combining detection tracks work. Even with perfect, uninterrupted gameplay footage of the entire

field at once, player tracks are still bound to have holes in their tracking due to breaks in gameplay and players not always spending the entire game-time on the field.

6.2.4 Manually Created Tracks

Even with everything discussed about tracks thus far, there's still one more important piece to cover. We want our final tracks to cover all instances of a player being on screen for the track to be considered complete, but between missed detections, modifying tracks and even outright deleting tracks, there's plenty of room for gaps in the track which should actually have detections. To cover for this the user is able to manually create tracks by choosing the first frame they need a track on and an end frame before the tracking picks up again and then run a real-time object tracker to create the missing track. This can then be modified, deleted or merged like any automatically created track.

6.3 Front-End Application

The application needed will be mainly implemented with Python and as a combination of its frameworks. The goals of the front end would be to show gameplay and statistics in an efficient and accurate manner.

6.3.1 Standard Web Development Libraries

- **Flask:** Flask is a Python-based web development framework. Flask is considered a lightweight framework allowing for quick and simple deployment of web applications or API points. Flask is also well documented and supported proving it to be a top Python web framework.
- **Plotly:** Plotly is a graphing library that helps to visualize and create an interactive graph. With Dash, Plotly can be a powerhouse to dynamic analytic visuals.
    - **Express and Graph Object:** Two of the most essential modules to Plotly are Express and Plotly Graph Objects. These libraries allow for quick and simple creation of strong figures to show off data. The Graph Objects library contains a hierarchy of different Python Classes referring to instances of Graphs for a figure. Express comes with over 30 functions for creating different figures. All functions from the Plotly Express module return instances of plotly.graph_objects.Figure, so essentially every figure made in Plotly uses Graph Objects.
- **Dash** is a Python framework to help build web-based analytical applications. Dash uses a combination of Flask, Plotly, and React. Dash excels at visualization of many kinds of data in Python. Dash applications are created on 2 main pillars: Layout and Callbacks.
    - **Layout:** The layout of a Dash app consists of what the application looks like through a hierarchical tree of components. The layout is purely an abstraction around HTML, CSS, and JavaScript. The two main sets of components include dash_core_components, which are a set of basic custom components that has been created and maintained by Dash for

Dash Applications, and dash_html_components which are HTML based components that are translated for easy use on Dash.

- **React:** Dash also provides a framework that can convert React.js components into python classes for custom made web components made from React.

- **Callbacks:** A callback are Python functions that are automatically called by Dash when an input component's property is changed. A callback decorator requires both an "Input" and "Output" values. Following the declaration of a callback will be the function that uses the "Input" to declare what the "Output" of the callback will be.

- **States:** A state will allow passing values without having to fire a callback. Will essentially change the input without a hard update

- **Limitations**: Dash, being an extension of Flask, allows the ability to have Dash components in a Flask environment possible. Allowing the opposite (i.e., Flask into a Dash application) is also possible but not recommended as that will restrict control since we would be starting bottom-up in the hierarchy and would limit Dash's capabilities rather than Flask's.

```python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Hello World!"
```

```python
from dash import Dash
import dash_core_components as dcc
import dash_html_components as html


app = Dash(
    __name__,
    external_stylesheets=['/static/dist/css/style.css'],
    external_scripts=external_scripts,
    routes_pathname_prefix='/dash/'
)

app.layout = html.Div(id='example-div-element')


if __name__ == '__main__':
    app.run_server(debug=True)
```

Figure 1: Example Flask application comparing to Dash application

6.3.2 Other Imported Libraries

- **OpenCV**: OpenCV is a library that allows media manipulation. In the frontend, this will give the user straightforward viewing of the game footage. OpenCV will also help keep track of data needed for the backend.

6.3.3 Deployment

- **Docker:** Docker is a platform for building applications based on containers. Docker containers allow for isolating applications while sharing the same operating system kernel. Isolating applications allow each container to have their own dependencies, whether it be software or hardware resources.
    - **Docker Containers vs. Virtual Machines:**
        - Containers can be faster and less resource heavy than virtual machines.
        - Virtual machines provide high flexibility whereas Docker containers focus is on applications and their dependencies.

6.3.4 UI Design

Figma

We carried out our UI design process in Figma. The biggest advantage of this UI design tool is that it is completely collaborative. We can have different members of the team working from remote locations, and we'll still be working collaboratively. Everyone on the Figma project is able to see what we're doing in real time. Figma has an online version and an application. In both we can visualize our projects, although from the cell phone we can not edit.

Figma is 100% designed to work as a team. We can maintain all our elements through component libraries, ordering typographic sizes, colors, shapes. This will allow us to create a completely consistent web application, where each member of the project can find all the material available in a single file.

The free version of this design tool allows up to two designers per project. However, as a student you can verify your status for a free Education Team, and we were able to add more designers to the project.

We created simple designs for the dashboards that will be part of the web application(admin side) before we implemented those using Dash Plotly. We were able to identify the necessary components, and agree on the work-flow that would be executed by the user(admin side) of the application. We can see in the images below the UI design and components layout for the Video Trimming dashboard and the UI design for the Track Annotation View.

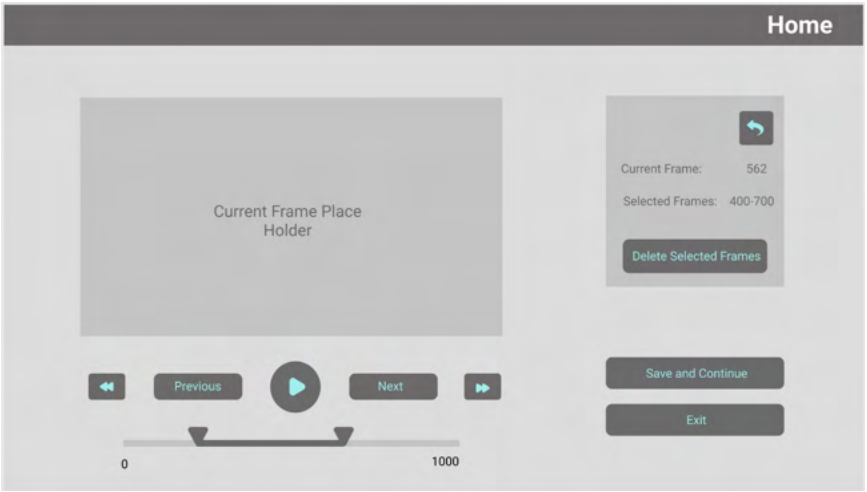


Figure 2: Figma UI Design for the Video Trimming Dashboard

Figure 3: Figma UI Design for the Track Annotation Dashboard

6.3.5 Video Cropping (Pre Processing)

This dashboard is part of the pre-processing that the submitted video will undergo before it is analyzed with the object detection and tracking algorithms. As we can see in Figure 2, the admin user will be able to select a number of frames using the range slider component (from dash core components) and delete those selected frames, to erase any footage that does not need to be analyzed.

The deletion of the frames will be carried out in the callbacks by labeling the selected frames(once the admin user decides to delete them) as *invalid frames*. The frames that will be included in the range slider interval will only be valid frames. Only valid frames will then be sent over for processing.

6.3.6 Track Modification (Post Processing)

This dashboard is for the admin user to work on annotating and modifying tracks after the video is analyzed with the player detection and tracking algorithms. On Figure 3, we can see that there is a *Tracks* card/column and a *Players* card/column, that will toggle between different views.

Track Views:

- *All Tracks*: This will show a list of all the unique tracks produced by the algorithm and the manually created ones. The item keys and values produced or created will be stored on the database and then imported to the dashboard in a pandas data frame called *detections*. That will be the data that will populate these views Each item(track) on the list will expand to show:
    - The track ID number
    - The start and end frame number for that track
    - *Go to Start* button that displays the start frame for that track
    - *Go to End* button that displays the end frame for that track
    - *Hide/Unhide Overlay* button that will show or hide the bounding box of that track on the frames.
- *Viewable Tracks*: This view will display a list of all the unique tracks on the current frame. It will retrieve all unique tracks that have that frame number from the dataframe. Each item(track)on the list will expand to the same data and button components in *All Tracks.*

- *Player Tracks*: This view will display a list of all the tracks assigned to a selected player. If no player has been selected it will just prompt the user to *Select a player to view tracks.* The list will keep the same format as in the two previously described views, with all items expanding the data and button components as in *All Tracks* and *View Tracks.*

Player's Card Column:

The two different player views will toggle between the home and visiting teams (A and B as we can see on Figure 3) on the match that is being analyzed. Both views will display a list of the team members on the match. Each item(player) on the list will expand the following data and components when selected.

- Player's Name
- Player's Jersey Number
    - Name data and Jersey Numbers for each team will be submitted when submitting the video for pre-processing and analysis.
- Player's Number of Tracks
    - It will increase every time a new track is assigned to a selected player, or a new track is created for a selected player
- *Add Selected Track* button
    - It will assign a selected track to that player
    - It will add the player's data to that track's ID on the data frame
- *Create a new track* button
    - It will allow the admin user to manually create a bounding box(new track) on a given frame and automatically assign it to that player
- *View Player Tracks* button
    - It will enable the *Player Tracks* view on the *Tracks* card/column to display all the tracks that have been assigned to that player. If no tracks have been assigned it will just show *No tracks have been assigned to this player yet*

Video Player Card:

The video player card will have the frame view component, buttons to toggle different settings and the video player component.

The buttons on the card header include:

- *Hide/Unhide all track overlays* button
    - It will either show or not the bounding boxes for all the tracks throughout the video.
- *Hide/Unhide Tracks Overlay in Scene* button
    - It will either show or not the bounding boxes for all the tracks on the current visible frame.

The Video Player component is running off a list of frames created from the uploaded video. This implementation is to help slow down the video and create more precise representation of the annotations provided from the models. The three essential dash components for the frame player are the graph, interval, and slider.

1. Graph: The graph shows the image using a figure created by plotly express module “imshow.” Updating the graph is done by providing the frame number to the imshow module and updating when needed.
2. dcc.Interval: The interval oversees keeping track of the frame to be shown to the user. Interval has four main properties:
    - interval: increments n_intervals every specified interval millisecond. Currently only able to update 250ms-500ms through local server-side callbacks.
    - max_intervals: the maximum number of intervals that can be called. Relates to the length of the list of frames stored in memory from the video.
    - n_intervals: the number of times intervals passed. This correlates to the frame number that will be provided to the figure.
    - disabled: will pause and play the interval. Created a ‘togglePlay’ callback that connects with a play button to control playing and pausing the frame player.
3. dcc.Slider: The slider works with the Interval to visualize the user where they are in the video as well as allows the user to go to specific points. The Slider’s value property correlates to the Interval’s n_intervals property.

The Video Player uses both Slider and Interval simultaneously so needing to synchronize them is needed in the update_figure callback. Making update_figure a **Circular Callback** allows Slider and Interval to be both the Inputs and Outputs, which also allows them to be in sync.

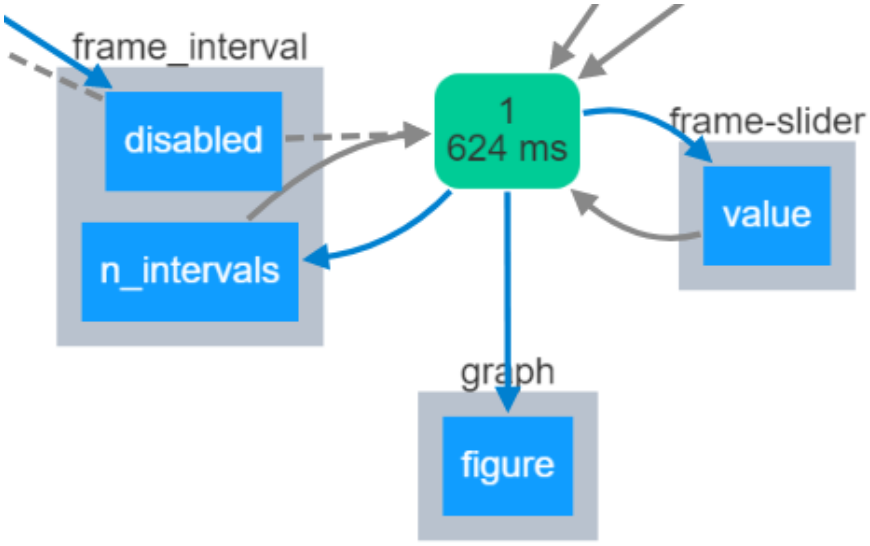


Figure 4: Circular Callback function recycling both Input and Output

6.3.7 Dashboard Implementation

Our Plotly Dashboards use dash html components, dash core components and dash bootstrap components.

- Dash core components allow us to insert high-level components into our dashboard.
    - We use the ***dcc.slider()*** component in order to scroll through the video frames and select an interval.
    - In order to display text we use the ***dcc.Markdown()*** components.
- Dash contains the HTML components library (dash_html_componentes) to be able to create HTML elements and be rendered by a web browser.
    - The way we compose our layout using Python structures with the components in this library.

    - One of the most used components in our dashboard is ***html.Div()*** to create different sections or blocks on the dashboard.
- Dash Bootstrap components for Plotly Dash make it easier to build consistently styled apps with complex, responsive layouts.
    - We use the ***dbc.Navbar()*** as well as ***dbc.Button(), dbc.Card()*** and ***dbc.Collapse()***.

The callbacks implemented make the dashboard interactive and connect the components. The definition callback decorator goes above the function indicating Input (core input component) and Output (HTML div component where we want the changes to happen). We create unique ids for the different components. The callbacks receive id corresponding to the components as input.
We can have similar components with different ids become interactive using the same callback. This means a callback can have multiple inputs.

- Multiple inputs can be added within the Input() list in the callback decorator.
- It is necessary to add multiple function/callbacks if we want to have multiple outputs. Each function/callback decorator is associated with one output.

It is not a good programming practice to use global variables. Traditional Python global variables are not safe to use in Dash as they are not shared across processes.Dash Callbacks must never modify variables outside of their scope. [1]

In order to share data safely across multiple python processes or servers, we need to store the data somewhere that is accessible to each of the processes.

- There are three main places to store this data:
    - In the user's browser session via dcc.Store
    - On the disk (e.g. on a file or in a database)
    - In server-side memory shared across processes and servers like a Redis database. Dash Enterprise includes onboard, one-click Redis databases for this purpose.
- To save data in user's browser's session:

- Data has to be converted to a string like JSON or base64 encoded binary data for storage and transport
- Data that is cached in this way will only be available in the user's current session.

6.3.8 Data Visualization (Client View) (FUTURE)

On our Web Standard Platform section we had listed plotly as a data visualization library. This library offers very simple implementations for the display of data. [2]. It becomes especially important when we aim to visualize a large data set or search for patterns in data. “Competitive sports data visualization is an increasingly important research direction in the field of information visualization. It is also an important basis for studying human behavioral patterns and activity habits.”[3]

Python is a general purpose language with powerful libraries for creating interactive visualizations and dashboards like Plotly and Dash. Plotly allows us to create endless types of interactive visualizations:

- Scatter plot: Display to compare correlation between 2 variables.
- Line plot: Visualization of the evolution of a variable.
- Bar Plot: Display the comparison between categories.
  - Generally, x-axis represents the categories and axis and value of those categories.
- Bubble Plot: Visualization similar to scatter plot where you can insert a third variable through the size of the marker.
  - Markers can be colored based on a fourth variable.
- Histogram Plot: Grouping the values of a variable into bins.
  - Provides variable distribution
- Treemap charts visualize hierarchical data using nested rectangles
  - In soccer this kind of visualization can be used to represent the players with most goals, assists or even penalization in a league since the data used is hierarchical
- Heat Maps: Visualization where the data is represented by color.

    - In soccer, heat maps are used as an indicator of the scope and frequency of a player’s movement. [4]

Dash allows us to create web applications using the core and HTML components. Dash makes use of Plotly to create interactive graphics in addition to other core components to turn our dashboard into powerful web applications with the possibility of multiple inputs and multiple exits.

Data Visualization Benefits:

- When we use graphics, we can pay much more attention to detail than in the traditional way. After all, there are images, trends, shapes, patterns and an entire universe in front of our eyes.
- When working with visual information, it is much easier to compare two trends. You can even put them in the same image, trying to understand how they differ. It's not hard to interpret because differences stand out quickly.
- If you had to do that using text, your job would be to compare all the information in one result with the other. This would certainly take time and could involve increasing the margin of error.

Techniques for Effective Data Visualization:

- Choose the right plot for a given dataset. We should consider the strengths and limitations of each chart type. Chart selection depends on the number of categories and measures (or dimensions) we want to display.
- Use different colors to highlight important information, but avoid making it too distracting.
- If we are analyzing high dimensional data it is really beneficial to have interactive plots so that we can zoom in the representation, hover over the figures and images or even have brushable charts so that we look at a section of the plot at a time to understand it better.

6.4 Amazon Web Services

Amazon Web Services (AWS) is Amazon's powerful cloud platform that provides many amenities to a developer. The products that are offered range over 200+ services from cloud computing to storage and databases which are needed for this application.

6.4.1 Computing

6.4.1.1 Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2) provides scalable cloud computing that allows the usage of all AWS resources through an instance of a virtual machine. The EC2 platform allows for the choice of an instance type, or a combination of:

- Operating System
- Processor
- Storage
- Networking
- etc.

Using EC2 has many advantages. One of the most convenient things about it is its flexibility. It is easy and relatively fast to rescale your project; in a few minutes, you can increase or decrease the capacity of your project. In addition, since some of the biggest companies in the world use EC2, it is highly reliable and offers all the support we need.

6.4.1.2 Elastic Beanstalk

Amazon Elastic Beanstalk provides a service for deploying and scaling web applications in the AWS Cloud. The process is as simple as uploading the code and Beanstalk will handle the deployment, capacity provisioning, load balancing, auto scaling, and application health monitoring. Beanstalk allows for complete resource control as you can maintain and select all the AWS resources needed powering the application.

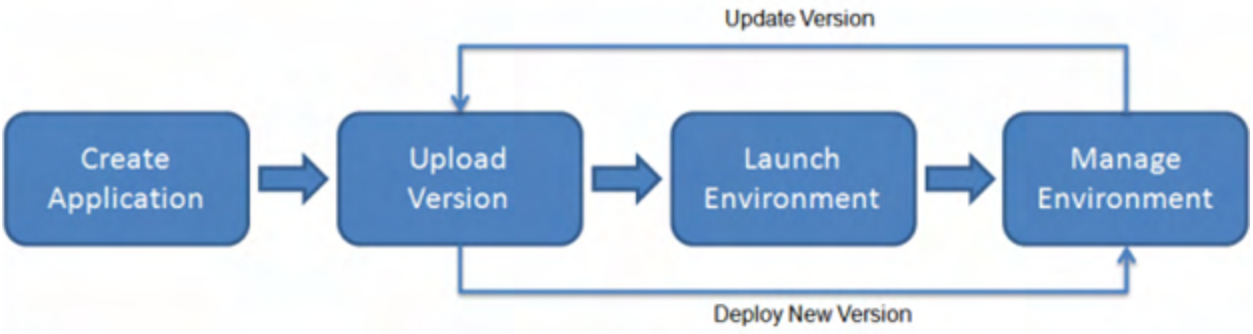


Figure 5: Beanstalk Deployment Flow

Creation of an environment includes the management and initialization of the following AWS resources:

- EC2 instance
- Load Balancer
- Auto Scaling Group
- S3 Bucket
- CloudWatch Alarms
- CloudFormation Stack

6.4.1.3 Elastic Container Service (ECS)

Amazon Elastic Container Service is a cloud computing service that manages applications through containers. These containers have three major components:

- **Task**: Blueprint describing how a container should launch
- **Service**: Enables running specified number of instances of a task definition simultaneously in an ECS cluster
- **Cluster**: Logical grouping of container instances

ECS allows specification of rules for different clusters on an EC2 instance. Container Services handle installation of containers, scaling, monitoring, and managing of all instances.

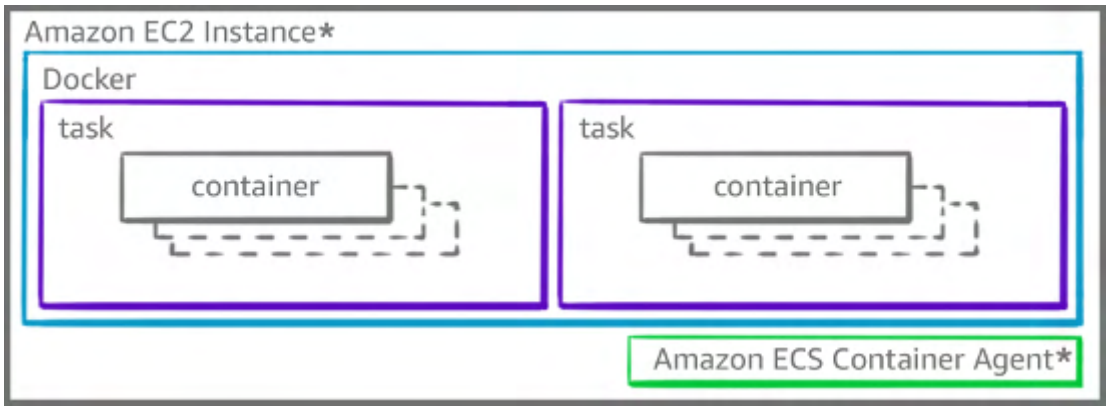


Figure 6: Elastic Container Service Example

6.4.1.4 Beanstalk vs. Container Service

Both Amazon Beanstalk and Amazon Container Service are able to handle the processes of containerized applications with an EC2 instance, but the two deciding factors come down to how many containers are needed to run the application and how much control is needed managing the application infrastructure.

The application can be built with either single container or multi-container structures in mind. When it comes to multi-container applications, Container Service is able to handle and specify each cluster independently while also giving more fine-tuned control for customizing each cluster's architecture. Beanstalk is also able to handle multi-container structures, but will handle the automatic provisioning.

With the current application's infrastructure having a single container, Beanstalk should help with simple deployment and automatic provisioning. If the application expands to multiple containers, migrating to Container Service should still be manageable.

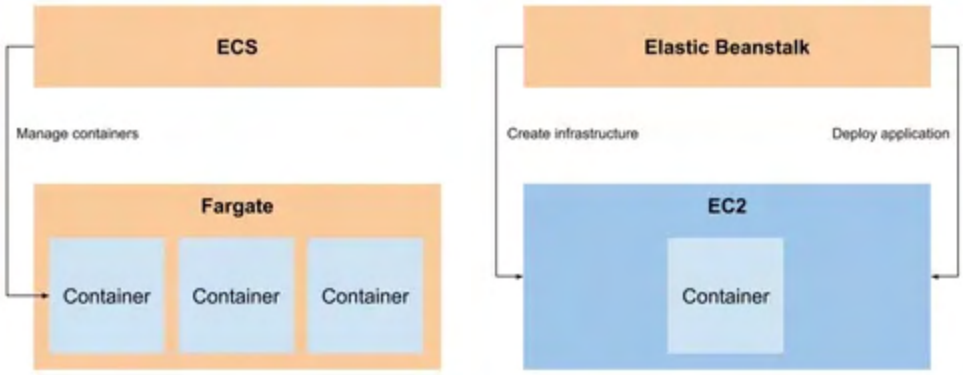


Figure 7: Elastic Container Service with Fargate and Elastic Beanstalk with Container

6.4.1.5 Lambda

Amazon Lambda allows for serverless automated computation. Lambda will be able to execute code in response to triggers. Lambda automatically allocates compute execution power to run code based on the incoming request or event. For this application, a trigger we can provide is when a video is uploaded to S3 which would then start the video processing in real-time.

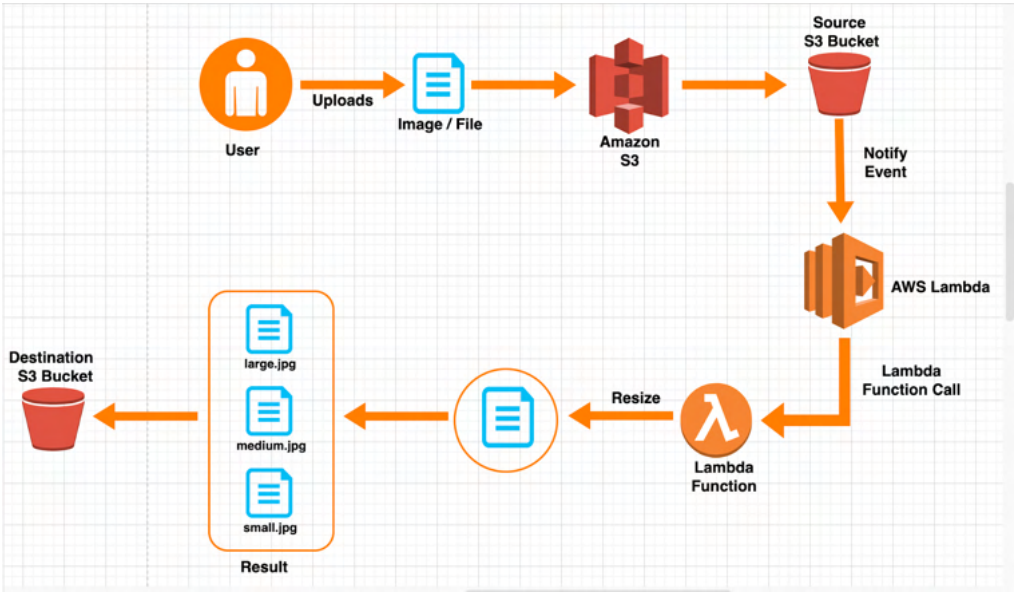


Figure 8: Example Lambda Use Case

6.4.2 Management

6.4.2.1 Identity and Access Management (IAM)

Amazon IAM is the gateway to all an AWS account can offer. IAM allows management to services and resources securely. This allows the ability to create and manage AWS users and groups by allowing permissions to certain AWS services.

This service will allow control to who can maintain the application and which users have specific access to certain AWS resources (e.g., Allow User A access to only the Database and EC2 instance, while allowing User B access to only the Sage Maker machine learning)

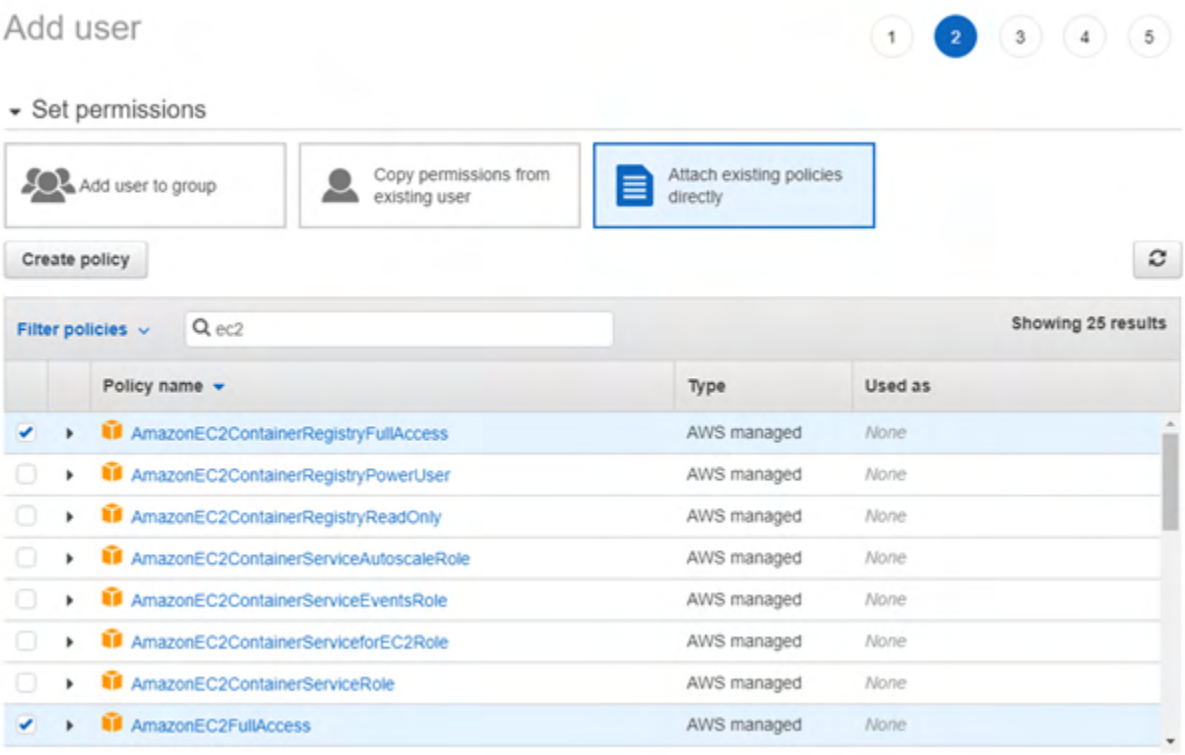


Figure 9: Permission adding in IAM Dashboard

6.4.2.2 Cognito

Amazon Cognito will allow for an easy and secure way for user access and authentication. Cognito will add the sign-up, sign-in portion of the application. Cognito will help show the users on the application while keeping them safe through its standard authentication support.

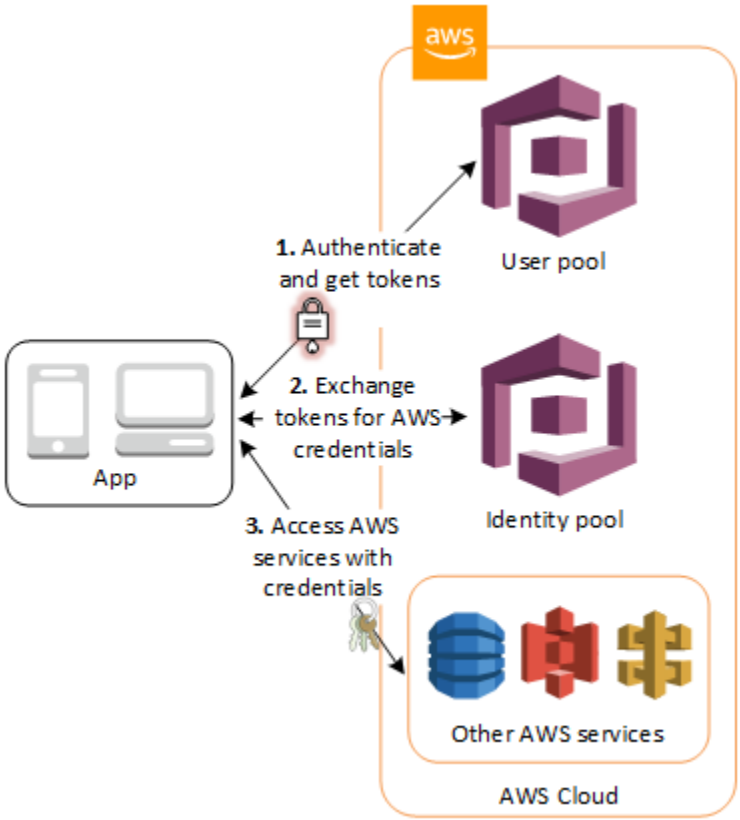


Figure 10: Cognito workflow

6.4.2.3 CloudWatch and Simple Notification Service(SNS)

Combination of Amazon's CloudWatch and Simple Notification Service will be a powerful tool to keep track of both billing and maintaining the many Amazon Web Services the application is using by providing the metrics of each AWS service. Alarms can be created in conjunction with SNS to keep in touch when certain metrics are at a monitored point. This service will allow easy monitoring to all the AWS services that are being used and will help with creating a price point.

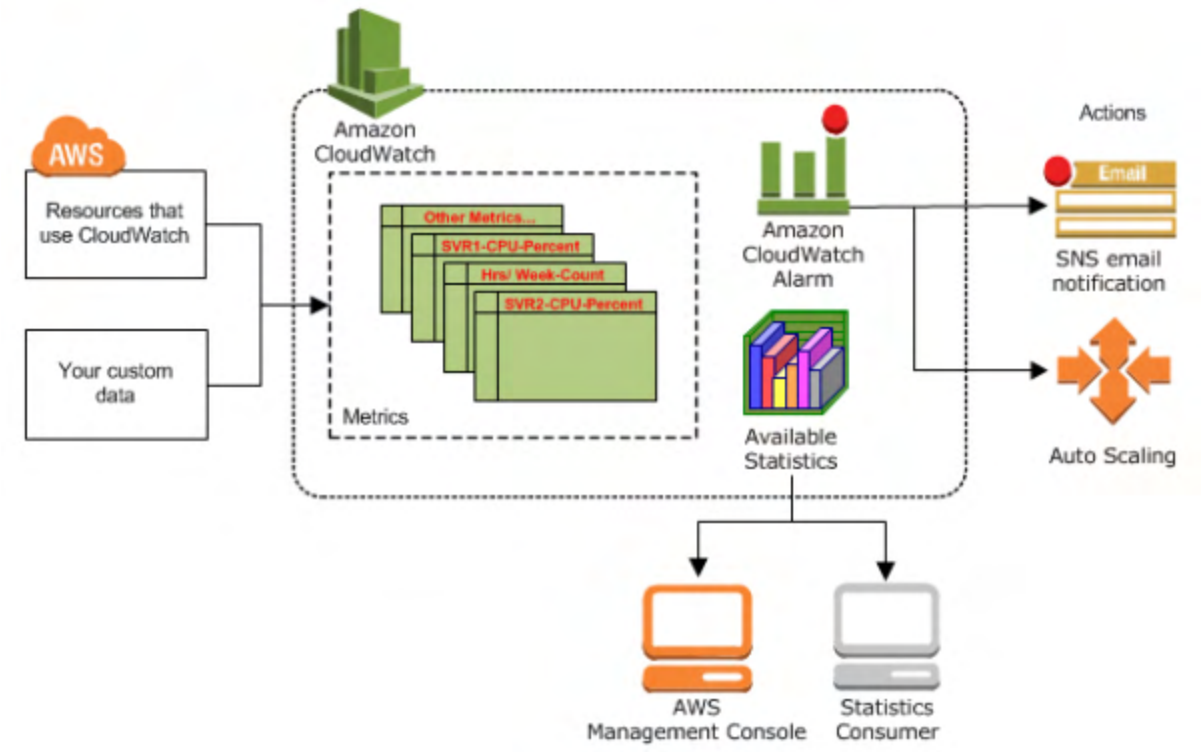


Figure 11: Example Use Case of Cloudwatch with Simple Notification Service

6.4.3 Storage

6.4.3.1 Amazon Relational Database Service (RDS)

Amazon RDS allows setting up and operating a relational database in the cloud. RDS supports an array of databases to store and organize data. Management tasks like migration, backup, recovery, and patching are also available.

6.4.3.2 Simple Storage Service (S3)

Amazon Simple Storage Service (S3) will allow secure, reliable storage of media from images to videos. S3 is designed to give developers an easier process to store their applications' infrastructure onto the cloud. S3 will allow for retrieving data from anywhere on the web.

S3 will have two main roles for the application. The application will need to store user uploaded videos, as well as retrieving the video for playback.

6.4.4 Machine Learning

6.4.4.1 SageMaker

Once we have all of our different machine learning models implemented on local machines, we need to start thinking about deployment of the pipeline to connect all of them and create an end-point for the front-end to use. As a group, we have decided to use AWS services, since they have a complete package of everything we need in one spot. Specifically for the purpose of deploying our machine learning models, we found that SageMaker under AWS might be the best option.

Amazon SageMaker allows for creation, training and deployment of machine-learning models in Amazon's cloud platform. SageMaker provides many libraries, like TensorFlow, to allow supported Machine Learning algorithms from scratch. For the application, a general workflow through AWS would look like:

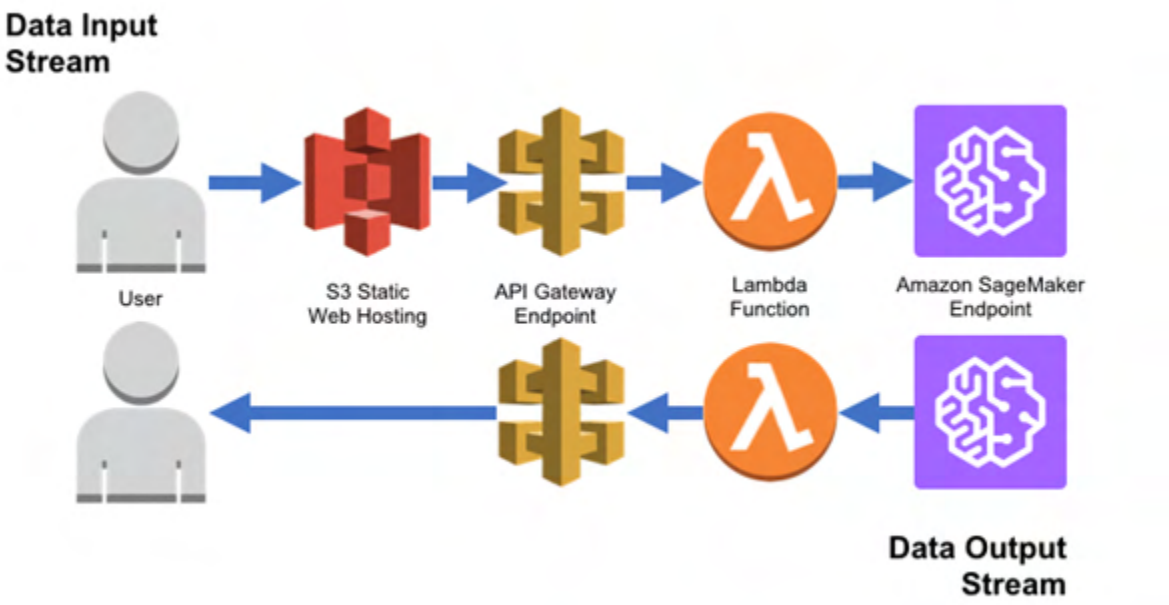


Figure 12: SageMaker workflow

Services

AWS SageMaker has a lot to offer. Some of the main services it provides are:

- Amazon SageMaker Ground Truth - a fully managed data labeling service that makes it easy to label all of your unlabeled data (a feature that we might use if we decide to train our own models instead of using pre-trained models)
- Amazon SageMaker Edge Manager - provides model management for edge devices so you can optimize, secure, monitor, and maintain your models.
- Amazon SageMaker Pricing Calculator - Since budget is important to us and we need to know what would be the cost of the services we are using, this calculator can save us a lot of time by providing us with the information of everything we will need to pay for.
- Different Computing Power - we still do not know how much computing power our models will require. Luckily, Amazon SageMaker gives us many option to choose from. We can choose different memory sizes and computing power for varying prices, as you can see in the image below.

Standard Instances	vCPU	Memory	Price per Hour
ml.t3.medium	2	4 GiB	$0.05
ml.t3.large	2	8 GiB	$0.10
ml.t3.xlarge	4	16 GiB	$0.20
ml.t3.2xlarge	8	32 GiB	$0.399
ml.m5.large	2	8 GiB	$0.115
ml.m5.xlarge	4	16 GiB	$0.23
ml.m5.2xlarge	8	32 GiB	$0.461
ml.m5.4xlarge	16	64 GiB	$0.922
ml.m5.8xlarge	32	128 GiB	$1.843
ml.m5.12xlarge	48	192 GiB	$2.765
ml.m5.16xlarge	64	256 GiB	$3.686
ml.m5.24xlarge	96	384 GiB	$5.53

Figure 13: Different memory and computing power

Pricing

Such great and comfortable services come with a price. Although Amazon AWS will charge you for using SageMaker, one of the nice things about it is that you only pay for what you have used. There are no minimum fees and no upfront commitments or payment. In addition, for the first 2 months of use Amazon will provide us with some free computing power; the image below shows the free services they will provide.

Amazon SageMaker capability	Free Tier usage per month for the first 2 months
Amazon SageMaker Studio notebooks , On-demand notebook instances	250 hours of ml.t3.medium instance on Studio notebooks **OR** 250 hours of ml.t2 medium instance or ml.t3.medium instance on on-demand notebook instances
Amazon SageMaker Data Wrangler	25 hours of ml.m5.4xlarge instance
Amazon SageMaker Feature Store	10M write units, 10M read units, 25 GB storage
Training	50 hours of m4.xlarge or m5.xlarge instances
Inference	125 hours of m4.xlarge or m5.xlarge instances

Figure 14: Amazon free computing power

6.5 Database

When choosing a database the primary choice to make is between a relational database versus a non-relational database. Here we will discuss the two and their primary differences, explain why we decided to go with a relational database over a non-relational database, as well as discuss some of the primary options in relational databases.

6.5.1 Relational Databases

A relational database, often referred to as an SQL database, is best thought of as a series of tables with direct relations to other tables through use of key values. A relational database is structured through multiple key parts, but the defining feature is how a relational database does its data structuring and relations. The table, or data structure model, acts as a completely rigid definition of data representation that cannot be broken short of changing the actual model itself [5, 6].

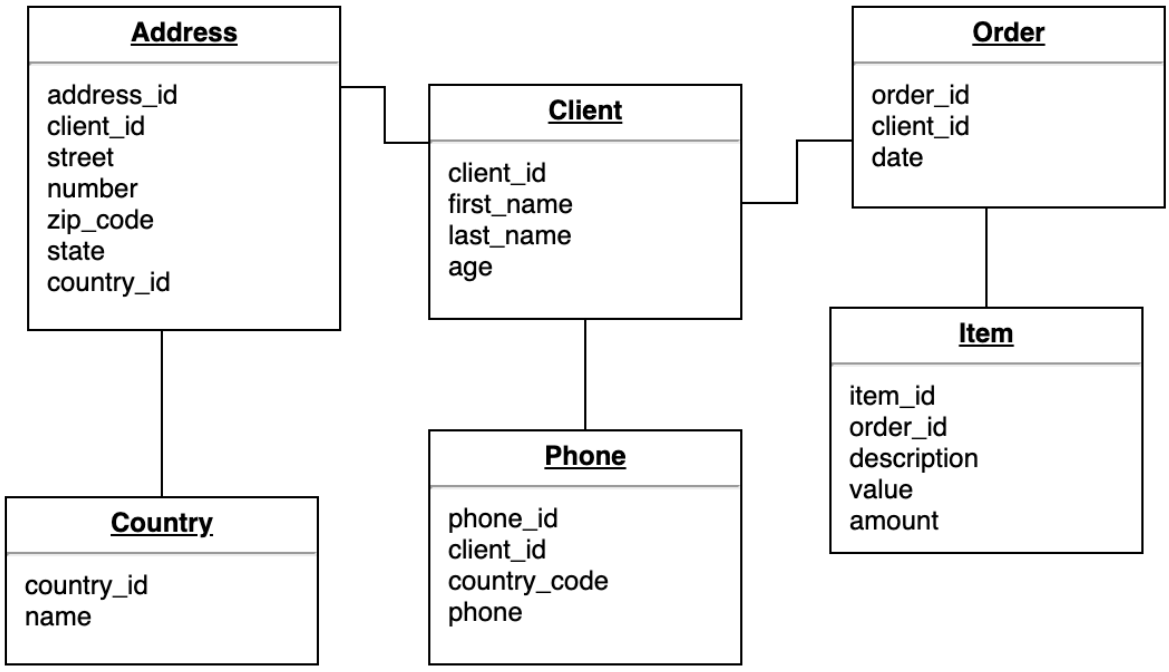


Figure X: Database Relations Example [6]

That makes a relational database a bit of a double edged sword. Because the database is designed around a rigid data structure it can make it very hard to change the design or add new columns to store information, however, the enforced structure means that the integrity of the data stored there will be of a higher quality and manipulating the stored data is very straightforward.

When discussing relational databases each row can also be referred to as a tuple which in turn effectively represents one entry or item stored within the table. The columns are each an attribute of the entries and the data type stored there is the domain [5]. In addition, the tables themselves are actually referred to as relations because the name of the table describes how it relates to the other tables stored in the database [7]. Lastly, each tuple stored in the table has a piece of identifying data, or ID, that is unique to that specific item.

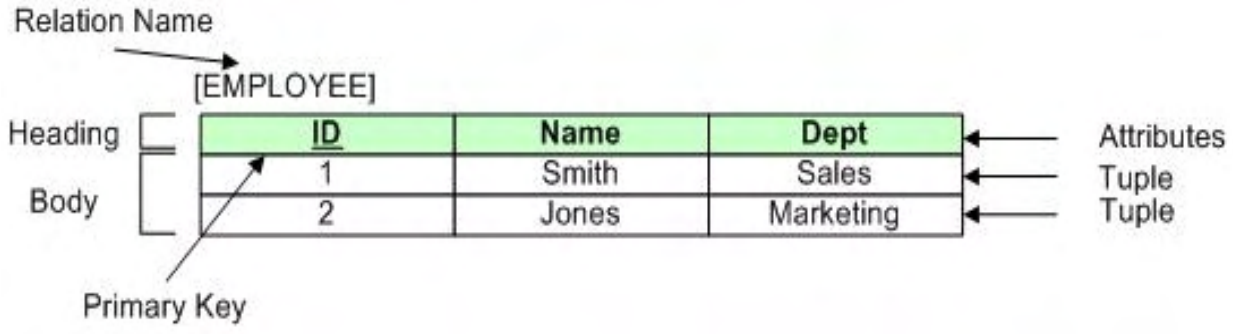


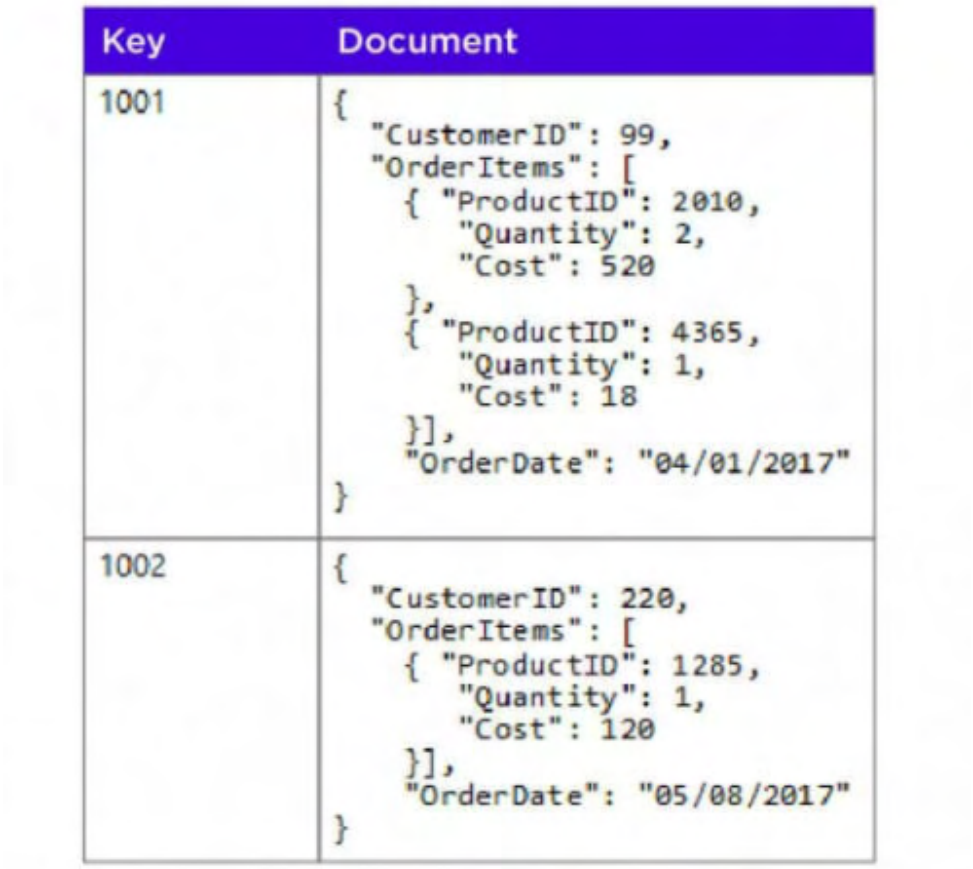
Figure 15: Relational Database Table Example [5]

6.5.2 Non-Relational Databases

A non-relational database, often referred to as a NoSQL database, relies on dynamic schemas that loosely defines the structure of the data that it holds [8]. Whereas a relational database is incredibly structured and all data stored within a relational table must be formatted in the exact same way, a non-relational database is much more relaxed in it's structure. A non-relational database allows the designer to include or disclude certain categories on an entry by entry basis, whereas in a relational database all categories must have data and you can't store an extra piece of data in an entry short of making an entirely new column for it.

When it comes down to it, the primary draw of a non-relational database is its flexibility. Rather than storing pieces of data in rows of a table, a non-relational database instead opts for a document-like structure. These documents can contain a wealth of detailed information and data all the while being able to be quickly and easily modified and

updated with new bits of data without any of the hassle of a rigid table structure. This makes them vastly superior when it comes to storing a lot of varied data and data formats because everything about an object can always still be stored in the same document [9, 10].



Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Figure 16: Example Document Structure [10]

So when it comes down to it each entry or tuple stored in a relational database row can be directly related to an individual document stored within a non-relational database. The primary difference is that the tuple of a relational database is inflexible and bound by the structure of the table whereas a document stored in a non-relational database is unbound by which collection of documents it's a part of. Individual documents can be modified without needing to modify the attributes of every other document stored within the collection.

6.5.3 Why Relational over Non-Relational Database?

After researching which style of database to choose the one we decided to go with was a relational database.

The reason to use a relational database would be if you require a database that's to be highly flexible in how it processes and stores said data and can easily adapt to changes in formats and data types. However, that is not of the utmost importance to our project. Instead we actually highly value the relational aspect of a relational database and the rigidity of the stored information. Every entry to our tables should always be of the exact same format

In addition we plan on storing a lot of entries in certain tables of the database. Each processed frame can include a dozen or more detections and games often have tens if not hundreds of thousands of frames, resulting in a million or more stored detections over the course of an entire game. This means that scaling our database vertically is much more important to us than being able to scale it horizontally, and relational databases are much more effective at working at that kind of scale [8].

6.5.4 Amazon Relational Database Services (RDS)

Because we plan on deploying through Amazon Web Services (AWS) we have to choose from the various AWS services provided for relational databases. Fortunately, AWS offers a host of different relational databases to choose from, including MySQL and PostgreSQL [11]. Both are remarkably similar, however there are some key differences to explore when deciding between the two.

6.5.5 MySQL

MySQL is the standard when it comes to relational, SQL databases. It's so widely used it's practically synonymous with SQL and it's even built into the popular LAMP development stack [12]. MySQL is very simple and easy to use, but outside of the basics of a relational database it doesn't have much else going for it. It's also optimized for speed, reliability and working at scale.

6.5.6 PostgreSQL

When it comes down to it, PostgreSQL can do pretty much everything that MySQL can do and more. PostgreSQL can handle more complex queries and even has support for

NoSQL data types. However, the most important difference between MySQL and PostgreSQL that really pushes the latter over the top for our uses is its ability to handle extremely large amounts of data with relative ease [12].

PostgreSQL is also ACID compliant, which means that it guards against data becoming corrupted through transactions. In addition, PostgreSQL also directly supports python, meaning that since that is our project's primary coding language, it will be a lot easier to integrate PostgreSQL into our code than MySQL.

All-in-all PostgreSQL is the clear choice between for our purposes when choosing a relational database offered through AWS.

6.5.7 Connecting Database to Dashboard

Module Psycopg2

We have integrated Postgres with Python(used in our Dashboard) using the psycopg2 module. Psycopg2 is a Postgres database adapter for Python. To use this module we followed this steps:

1. Installing the Psycopg module with the pip command.
2. To connect to our database, we created a connection object that represents the database.
3. Next, we created a cursor object to help us execute your SQL declarations.
4. We connected to our database using the connect() function of the psycopg2 module.
5. The following is the list of parameters that we have passed to the connect() function:
    - database: The name of the database you need to connect to.
    - user: The user name to use for authentication.
    - password: The password of the user's database.
    - host: The server address of the database. For example, a domain name, "localhost", or IP address.
6. We can then load the contents of a table from the database into a panda dataframe

Pandas Dataframe

Pandas is a library for data analysis that has the data structures that we need to clean up raw data and that are suitable for analysis (e.g. tables). It is important to note here that, since pandas performs important tasks, such as aligning data for comparison, merging datasets, managing lost data, etc., it has become a very important library for processing high-level data in Python. In particular, it provides data structures and operations for manipulating numerical tables and time series. [13]

The library features are:

- The data type is DataFrame for data manipulation with built-in indexing. It has tools for reading and writing data between in-memory data structures and varied file formats
- Enables data alignment and integrated handling of failed data, dataset restructuring and segmentation, tag-based vertical segmentation, elegant indexing, and horizontal segmentation of large datasets, inserting and deleting columns into data structures.
- You can perform operation strings, split, apply, and merge over datasets, mixing, and joining data.
- Allows you to perform hierarchical axis indexing to work with high-dimensional data in smaller data structures, time series functionality: date range generation and frequency conversion, linear regression and statistical window scrolling, date offset, and delays.

Pandas has functionality that allows us to read and write information in relational databases. A relational database is a set of one or more tables structured into rows (records) and columns, linked together by a common field, called a key. This way of building databases is called a relational model.

The structure of the tables is analogous to the data frame concept. It is essential that each record contains a unique identifier, which will be used as the primary key to explicitly access that record.

The term query defines the action of getting data from the DB.The panda library allows us to run a query on a single line, without having to connect to the engine and close the connection explicitly.

This is achieved with the read_sql_query() function, which collects as the first argument a string with the query text, and as a second argument the engine with which we want to connect. The result of the query will be a data frame, which we store in a variable.

6.6 Object Detection

One of our main project requirements is to implement a player detection algorithm that will draw on the input video bounding boxes for the player detected and output the detected frames so that the players can be tracked in the same analysis process and later using real time tracking. See picture below for an example.

Figure 17: Bounding Boxes Implementation Example

Our first step consisted in trying out different algorithms and model implementations to determine which one would produce the closest to the desired results, to later custom train our models.

6.6.1 Object Detection: Using Tensorflow Object API, pre-trained and custom trained models.

To get started, it's important to know the distinction between an object classifier and an object detector:

- Image Classification: Algorithm that is responsible for classifying a whole image within a category. For example, we give it a photo of a car and just it mentions that the predominant object in the photo is a car.

- Object Detection: An algorithm that detects multiple elements within an image and classifies them. For example, we give it an image that contains a traffic stop and it identifies that in the image there are a traffic light, cars, traffic signals and trees.

The preliminary implementation for the player detection used Tensorflow Object Detection API. TensorFlow is one of the most important Deep Learning platforms in the world. This open-source development of Google goes beyond Artificial Intelligence, but its flexibility and large developer community has positioned it as one of the leading tools in the Deep Learning sector.

Its object detection API makes it easy to build, implement, and train object detection models. It provides pre -trained object detection models for users running inference jobs. We do not need to train models from scratch.

We first worked with a custom trained model built to detect players and the soccer ball. About 50 frames were extracted from a soccer match video, and the desired detections were annotated to these frames. The annotated frames were then used to feed the model selected (Faster RCNN Inception Model from the Object Detection Zoo) with the purpose of training it.

We do training in the following steps:

1. Preparing the data
    a. Prepare our data for training, i.e. mark in a training set the coordinates where the objects we want it to detect are.
    b. We use **labelImg** a program that helps us to do this easily and gives us an XML file with the information we require.
2. Convert the XML data to TFRecord. TFrecord is the image format that our algorithm needs to be able to train.
3. Training
    a. Choose the model we will train.
    b. Prepare files for training
        i. Model configuration
        ii. Training Labels
        iii. Computational graph of the model
    c. Train the model

The label map tells the trainer what each object is by defining the class name mapping to the class ID number. In the case of our implementation we have only defined the player and ball labels which is the list of strings that are used to add the correct label for each bounding box that surrounds a detection. In this file we tell our algorithm which labels we will train it on. The name we put on the labels must be the same name that we use in the **labelImg** tool (the package we are using for labeling). Basically this file has a series of 'item' elements with their respective identifier 'id' and class name 'name'. Here's an example, this changes based on the number of items you want to learn how to detect.

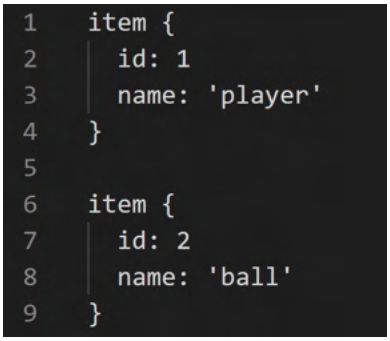


Figure 18: Data Labeling Example

We configure our training and it the .config file we have what our training script will read to know extremely important parameters, such as:

- Where to get the tfrecords.
- Where to get the tag files label_map.pbtxt.
- Where to find the required files of our model (checkpoints).
- The number of steps to train.
- Batch_size (Number of images that we will train in each iteration, we can start with a low number like 1 and go up if we see that our computer supports it)

When We are training the model what we are looking for is to reach a very low 'loss', at least it is below 0.9. After we train we get our graph.pbtxt, this file is the one that contains the information necessary to be able to make predictions(detections) later, but before this we have to 'freeze' our model, that is, we will convert our checkpoints to a final model.

Training using different videos with multiple camera angles is needed for better results. For example if we only work with high quality images and then we run the algorithm using a blurry video as input we were not able to detect the ball in most cases. The blue

square detects the area where there is the most action in the field like we can see in the following images.

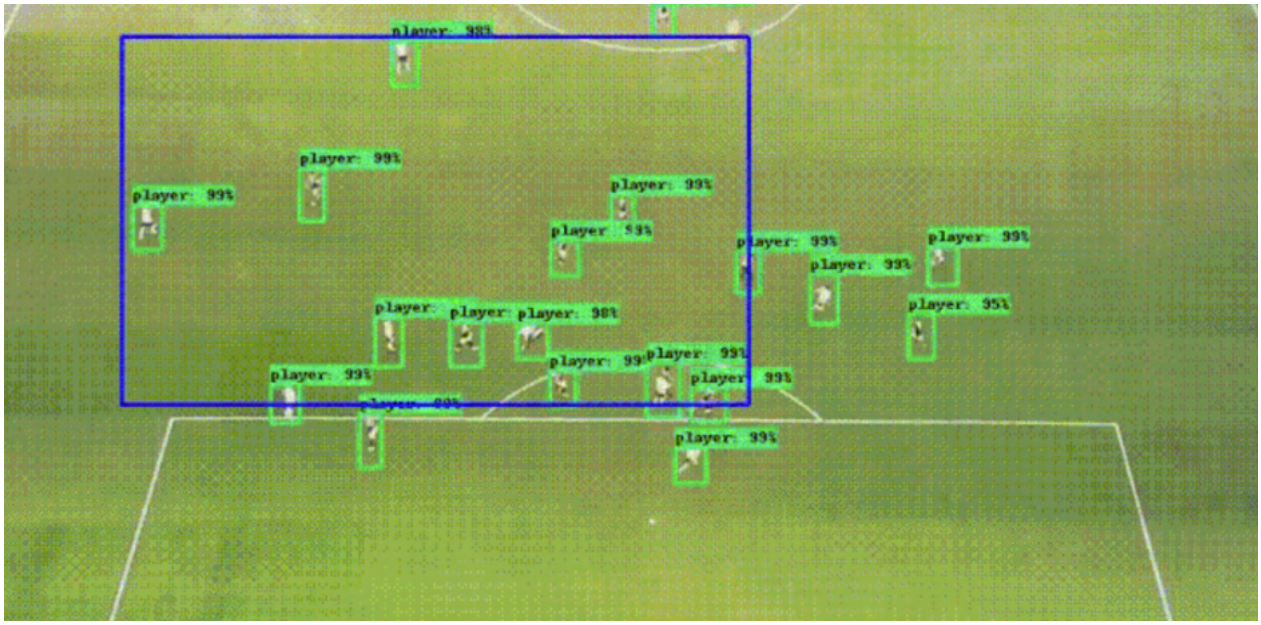


Figure 19: Most action area detection. Blue bounding box.

The implementation we have used so far loads the model by setting the pathath to a frozen detection graph, which is the actual model that is used for the object detection, it loads the tensorflow model to memory and it loads the previously mentioned label map.

Then the model is run on the input video and extracts info from object detection:

- Each box represents a part of the image where a particular object was detected.
- Each score represents the level of confidence for each of the objects. Score is shown on the result image, together with the class label.
- Then we visualize the results of the detections.
- After we draw the boxes on the output video.

Like we mention above, the information about the classes, frames, and scores of each frame passed to the object detection is then obtained. The results can be filtered with respect to their score, in this case we are particularly interested in those with high scores.

Now we need to figure out the area of the frame with most players present and where the ball is located. We assume this is the area where it is most likely to be the most action area. We then continue to draw the rectangle that limits/highlights the area:

- We first want to make sure we fix the area of the rectangle to have smaller dimensions than the actual frame obtained from the video, in order to account for execution time.
- Once we have the dimension of the chosen area, we iterate over the entire framework by systematically choosing many areas and calculating how many players are in them.
- Select the area with the highest score and add that rectangle in the frame

We tried different pre-trained models with the Object Detection API to determine which one would obtain the most optimum results. The models that we implemented were pre-trained using the COCO dataset, which recognizes 80 different object categories. For example, person, car, tree, building.

We were interested in the person category (for players) so we implemented the models in the following table. The COCO mAP column shows the model's accuracy index. Higher numbers indicate better accuracy. As speed increases, accuracy decreases.

Model name	Speed	COCO mAP
`ssd_mobilenet_v1_coco`	fast	21
ssd_inception_v2_coco	fast	24
faster_rcnn_resnet50_coco	medium	30
faster_rcnn_resnet101_coco	medium	32
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37

6.6.2 Results

We started with ssd_mobilenet_v1_coco and did not achieve desired results during testing, when there were occasions when we were only able to detect less than 50% of the players present on the frame.

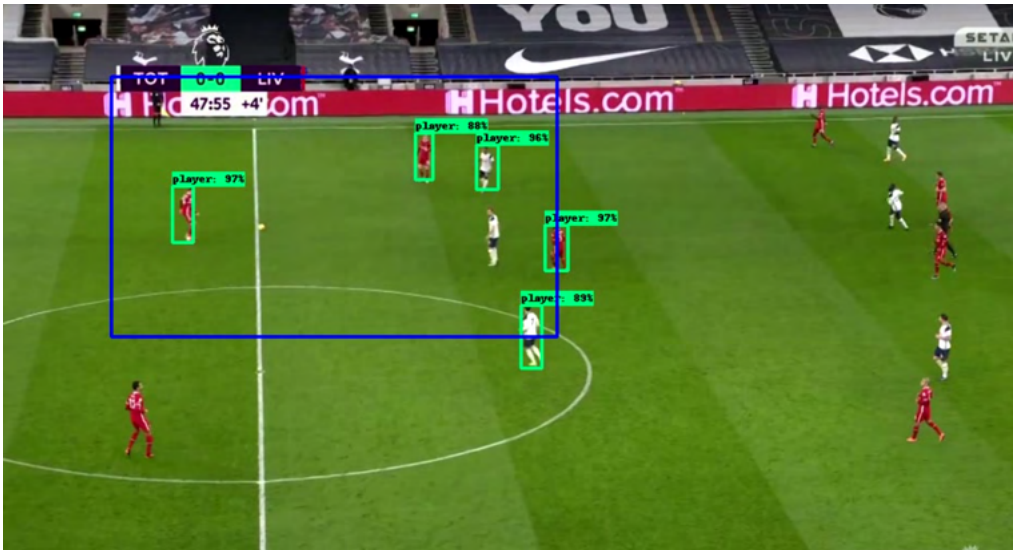


Figure 20: Results for ssd_mobilenet_v1_coco

The best results that we have obtained so far using this algorithm have been while loading the faster_rcnn_resnet50_coco model. We have been able to detect almost 100% of the players all the time with the testing data that we have been using. See the image below to compare results.

This algorithm implementation not only detects the players and the ball but it also attempts to predict the area with the most action in the frame, which is represented by the blue bounding square.

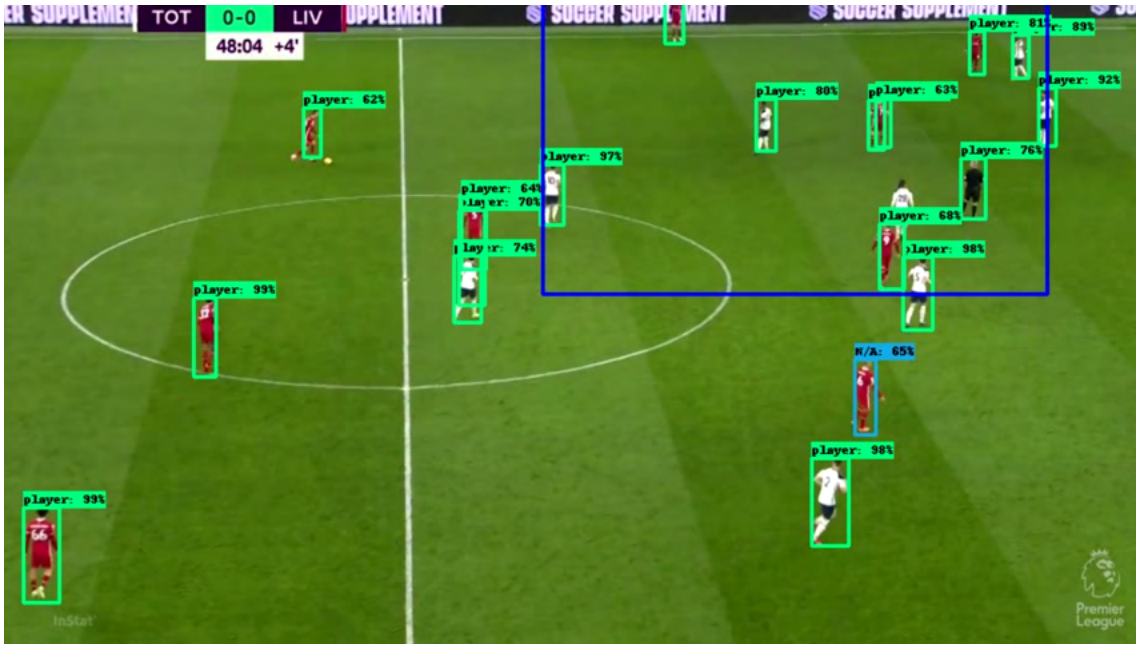


Figure 21: Results for faster_rcnn_resnet50_coco

6.6.3 Future Work

We would like to try different models and custom train them using a more complete data set that includes different quality fractions of video frames, with the player and ball data that we are interested in, including different training models. This process includes finding the data, data labeling and model training.

Another next step will be testing the different models implemented using longer videos that feature different camera angles and lighting. This will then help the team get a more real idea of the results the algorithm is able to produce and what we would need to update or improve.

The integration of the object detection and object tracking algorithms is still pending until they are both optimized. They will be the first part of the processing after the video is submitted to the Sports Science Team and it is edited by them using the video trimming dashboard.

6.6.4 Object Detection using Yolov3 and OpenCV

Something that we were able to observe when we were testing the pre-trained object detection tensorflow models is that we would not get results as fast as we would if we implemented object detection with Yolov3.

One of the best algorithms for processing images in real time is YOLO. It comes from "You Only Look Once". YOLO uses deep learning and CNNs to detect objects, and distinguishes itself from its "competitors" because, as its name implies, it requires "seeing" the image only once, allowing it to be fast(although it sacrifices a little accuracy). This speed allows you to easily detect objects in real time in videos (up to 30 FPS).

How YOLO performs detection:

- It first splits the image into an SxS grid (image on the left). In each of the cells it predicts N possible "bounding boxes" and calculates the level of certainty (or probability) of each of them, that is, SxSxN different boxes are calculated, the vast majority of them with a very low level of certainty.
- After obtaining these predictions, it proceeds to remove boxes that are below a limit. The remaining boxes are given a "non-max suppression" step that serves to eliminate possible objects that were detected in duplicate and thus leave only the most accurate of them.

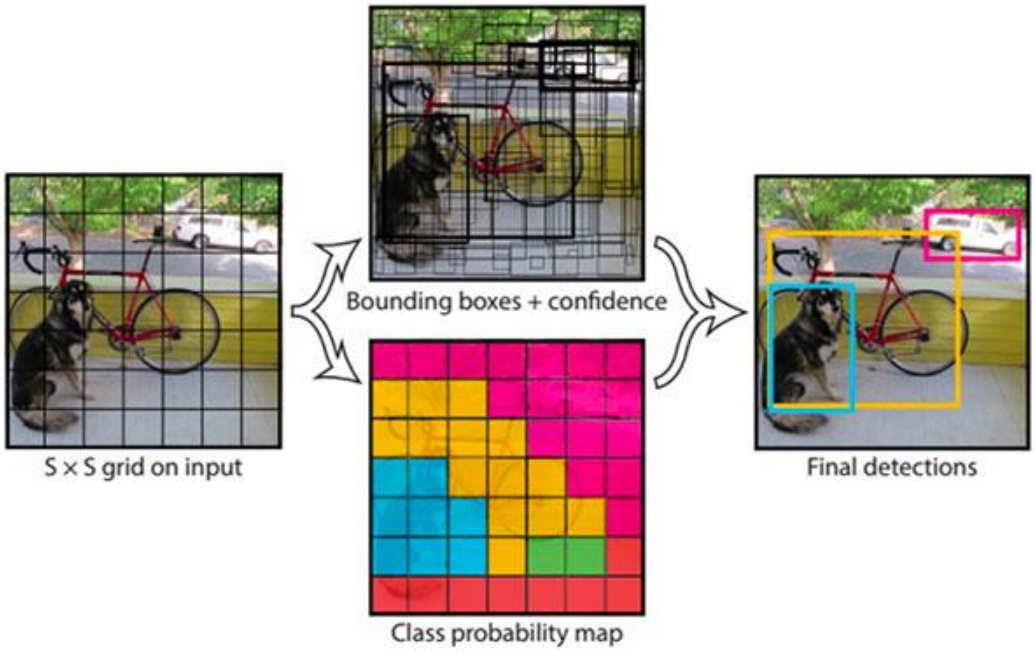


Figure 22: How YOLO performs detection

YOLO can have different implementations. The Deep Dark Net (the one we have worked on) is the "official" implementation of YOLO. It is written in C with CUDA, so it

supports GPU computation. The downside is that, because it is not based on an extended-use framework, it might be more difficult to find solutions to possible errors. Also the process for training the model may not be as smooth, and it might take somewhat of a learning curve on our side.

Our YOLO Implementation:

- The YOLO model we use has been pre-trained in the COCO dataset that we mentioned in the previous section.
- Loads the COCO class labels our YOLO model was trained on.
- Initialize a list of colors to represent each possible class label. And that is why we get the different players in different colors.
- Derives the paths to the YOLO weights and model configuration.
- Loads the YOLO object detector and determines only the output layer names that we need from YOLO.
- Loads the image(video) and sends it through the network.
- Performs a forward pass through the YOLO network.
- Show the inference time for YOLO.
- Visualize and filter unwanted detections.
    - Extracts the class ID and confidence (i.e., probability) of the current object detection.
    - Filters out weak predictions by ensuring the detected probability is greater than the minimum probability.
    - Scales the bounding box coordinates back relative to the size of the image.
- Writes the output frame.

6.6.5 Results

In the images we can see that we get 100% of players detected for our test video. The second image shows how it even detects with accuracy frames that present difficulties caused by camera movements and fast moving objects like the ball and players. It detects them in a fast shot and camera pan, when the game moves from the midfield to the goal area.

Figure 23: Results for YOLO implementation (1)

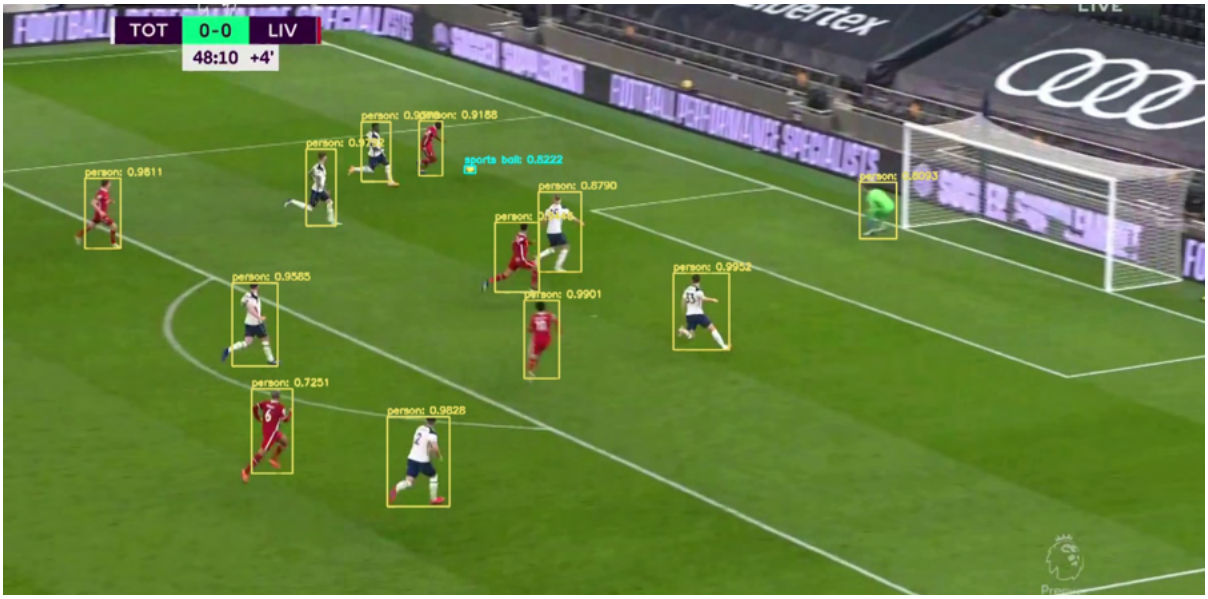


Figure 24: Results for YOLO implementation (2)

We were able to get a really fast implementation of YOLO for object detection. Even though we have gotten better results using this model for our testing data research shows that YOLO has known drawbacks:

- It does not always handle small objects well.
- It especially does not handle objects grouped close together.

The reasons behind these limitations are because of the actual way in which YOLO does object detection. It divides the input image into an SxS grid where each cell in the grid predicts only a single object.

If there exist multiple, small objects in a single cell then YOLO will be unable to detect them, ultimately leading to missed object detections. If our testing data includes small objects very close to each other we might not be able to visualize all of the desired detections.

6.6.6 Future

The next steps concerning this model will be attempting to pre-train it with images from a data set with soccer matches with different lighting quality and angles and extending the training data to be more varied as well. We want to make sure the YOLO model works with accuracy in the kind of videos outside users will be allowed to submit to the webportal for analysis.

6.7 Player Tracking

6.7.1 Multiple Object Tracking Explained

Multiple Object Tracking (MOT), or Multiple Target Tracking (MTT), plays an important role in computer vision. The task of MOT is largely divided into a few separate tasks:

- Locating multiple objects
- Saving their identities
- Following the objects throughout the input video

The above tasks of multiple object tracking additionally are eventually determining the number of objects in the video, which typically changes over time, and keeping their identities the same the whole time - which creating one of the most difficult issues multiple object tracking models face - what happens when an object gets out of the frame and then comes back.

Additional important problems that make things more difficult for multiple object tracking:

- Frequent occlusions
- Initialization and termination of tracks
- Similar appearance
- Interactions among multiple objects

Multiple object tracking is being used today to follow many different things:

- Pedestrians on the street
- Vehicles in the road
- Sport players on the court
- Groups of animals

It has a large number of important everyday applications:

- Visual surveillance
- Human computer interaction
- Virtual reality
- Stats in sports

Multiple object tracking can also be divided into two different categories: online tracking and offline tracking.

The difference between the two is that in offline tracking observations from future frames are being used when the model is processing the current frame, however, in online tracking, the model does not have any knowledge about future frames.

Online tracking can come in handy when the input is a live stream. Although online tracking does not use future knowledge, some online models show surprisingly good results when compared to offline tracking.

6.7.2 Multiple Object Tracking - Yolov3 + TensorFlow + DeepSort

In our project one of the first priorities was to get some kind of a result from tracking players, no matter the quality, so the next stages could be worked on.

As it can be seen in our block diagram (Section 6) - semantic segmentation, camera parameter estimation, and the web platform all depend on the results of the multiple object tracking.

After making some research, we have decided to implement our first model using Yolov3 for detection, and TensorFlow and DeepSort for tracking. DeepSort is a tracker that uses Kalman filters algorithm to try and have the best guess for the next track, using previous and current tracks.

We used a pre-trained model that was not trained on soccer data, however the results were better than we thought; Figure 25 below you can see an example of the model's output. The errors we had were mostly for changing the bounding box identification number when it's not needed, or losing a track because a player was covered with something else (ex. Another player next to him on the field); Figures 26-28 show examples of the different types of errors our model produces.

Output example:

Figure 25: Detections in one frame (left) and the bounding boxes coordinates (right)

Different Errors:

Here, if you look at detection number 10, you can see how two close people were detected as one person.

Figure 26: Error - one detection for two people

In the example below you can see how the player between detection number 8 and detection number 13 is undetected because he was in a close proximity to other players.

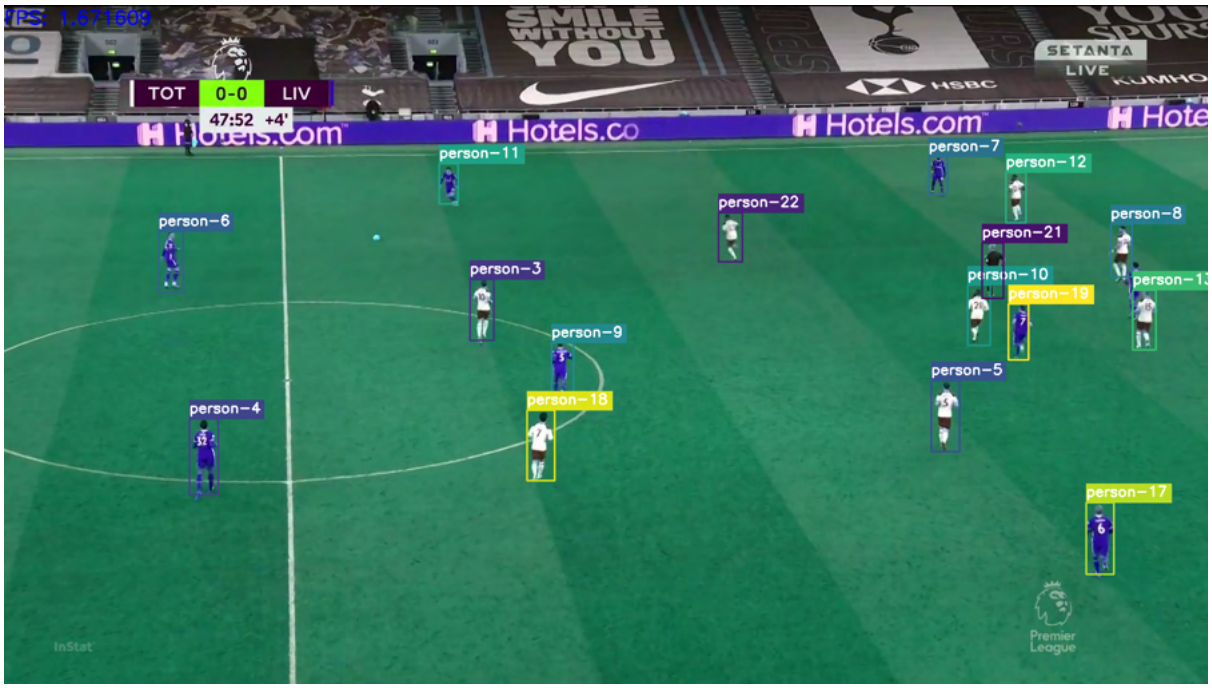


Figure 27: Error - player did not get detected

Lastly, in the following image you can see how detection number 69, has additional detection underneath it, meaning that player was detected twice.

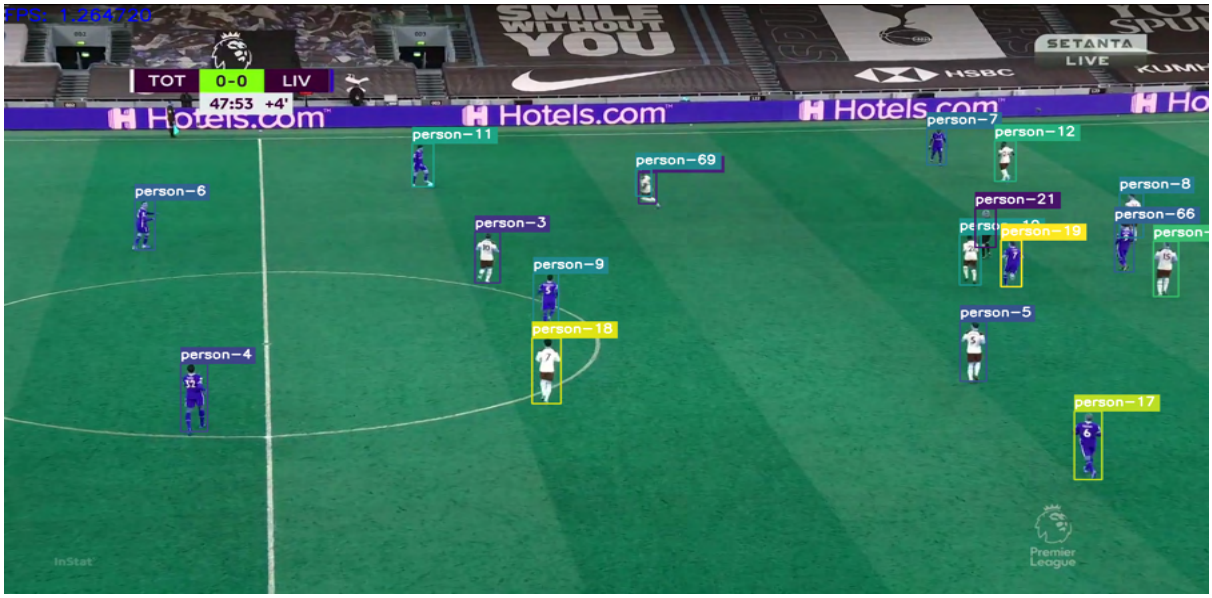


Figure 28: Error - two detections for one player

Efficiency:

Errors were minimal but the efficiency was bad, as it ran ~3fps. The input for the model is an mp4 video, and the output as an mp4 video with bounding boxes tracking players, each frame with the bounding boxes, a txt file with the [x, y, width, height], and the id of each bounding box for each frame.

Environment:

To run this model, we created a conda environment on a windows machine and an NVIDIA GeForce 940 graphic processor.

6.7.3 Multiple Object Tracking - Yolov4 + TensorFlow + DeepSort

Going a few weeks into the future, we needed to create some sort of a pipeline for the different models.

The first step was to go from detection and tracking into semantic segmentation and camera parameter estimation. Zeke and Brendan worked together to try and combine the results from detection and tracking into semantic segmentation and camera parameter estimation.

We had to get both models to work on one machine. Already having experience with the environment set up for new models, we knew it would not be an easy thing to do, and we were right. No matter what we did, we could not make Yolov3 work on the same machine as the homography models. Next, we tried to implement Yolov3 on the cloud, using Google Colab; for some reason, Yolov3 does not work on Google Co-Lab either.

After some research, we found out that there is a new Yolo version - Yolov4. We implemented Yolov4 + TensorFlow + DeepSort on Google colab and shared it with each other. Since the results (examples are shown below) were similar, and the speed of the new Yolo version is faster, we decided to stay with the new Yolov4 version.

Yolov4 uses the same pre-trained model that was not trained on soccer data, and the results are similar; in figure 29 we show an example of the output our model produces - a frame with a txt file showing the bounding boxes coordinates. The errors we had were mostly for changing the bounding box identification number when it’s not needed, or losing a track because a player was covered with something else (ex. Another player next to him on the field); Figures 30-32 demonstrate the different errors.

Output example:

Figure 29: Detections in one frame (left) and the bounding boxes coordinates (right)

Different Errors:

Here, if you look at detection number 3, you can see how two close people were detected as one person.

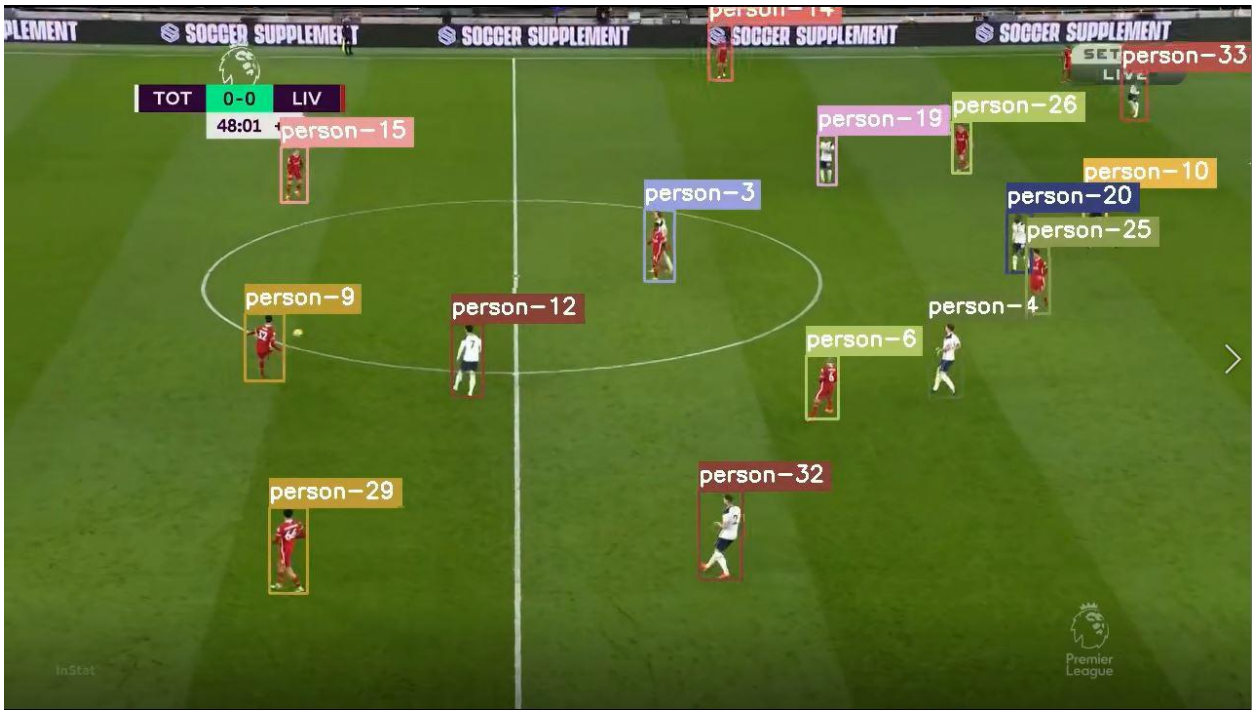


Figure 30: Error - one detection for two people

In the example below you can see how the players between detection number 39 and detection number 26 are undetected because they were in close proximity to each other.

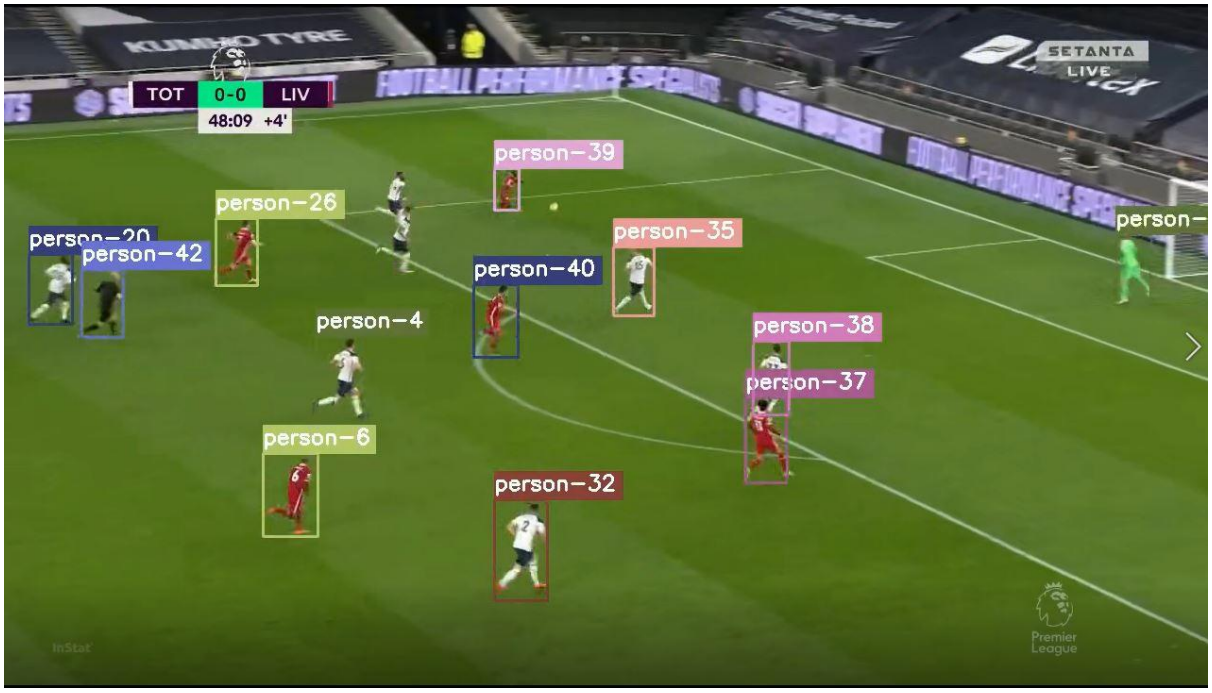


Figure 31: Error - players did not get detected

Lastly, in the following image you can see how detection number 2 should not be there. The model detected the whole screen as a person.

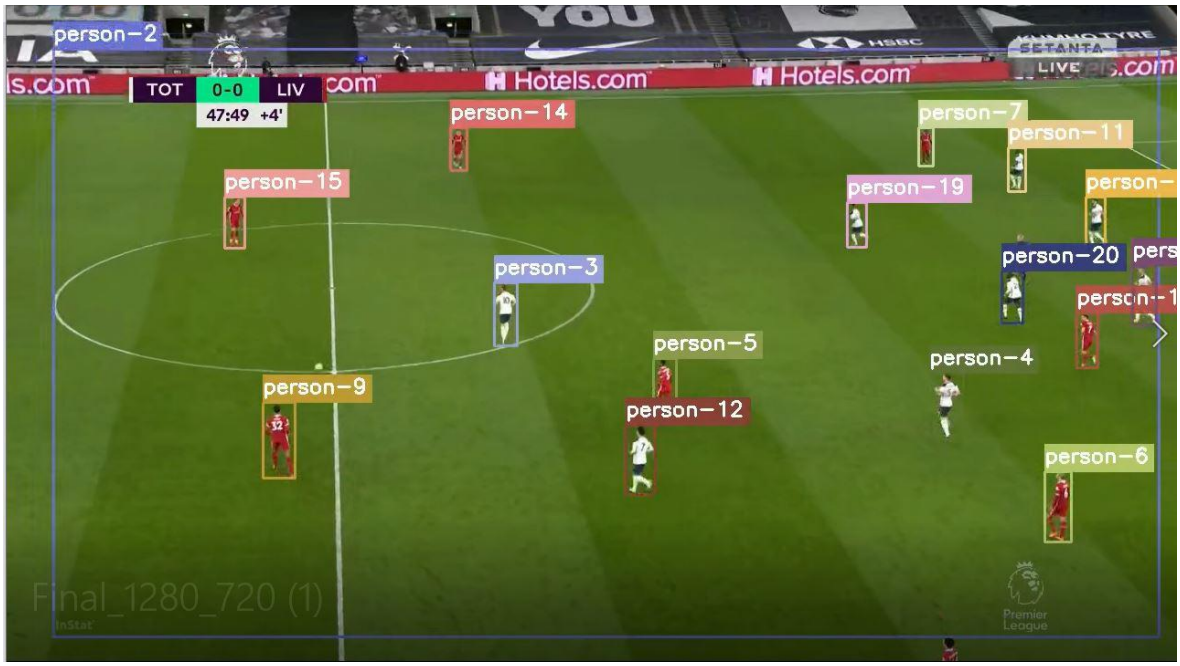


Figure 32: Error - screen detected as a player

Efficiency:

Errors were minimal in this model as well. To our surprise, the efficiency was decent, as it ran ~12fps. The input for the model is an mp4 video, and the output as an mp4 video with bounding boxes tracking players, each frame with the bounding boxes, a txt file with the [x, y, width, height], and the id of each bounding box for each frame.

Environment:

To run this model, we used the cloud. We implemented it on a Google Colab, utilizing the free GPU time they provide.

Future Use:

Looking at the final product we are trying to build, it is important that the different parts we create will work in harmony at the end.

Although other models might have performed better when we tested, this model was by far the easiest to implement. In addition, this model works the best with the models that come after in the pipeline.

Since the results of this model were completely useable, and it was easy to implement and work with the rest of the models in our project, we chose to use this one as our final model for detecting and multiple object tracking.

Others:

Since this model is the first to run in the pipeline, the output should match the input for the next model. Working together we figured that the best output would be as an mp4 file for the video, and a txt file for each fram with the coordinates in it. This again was not easy and we had to use a pip package called PythonVideoConverter. In addition, the required resolution for our model is 1280X720, and we used CV2 to convert the input video for this model.

6.7.4 Multiple Object Tracking - FairMOT

After we had our first model up, running, and producing decent results, we now expanded our research to find other models that we can implement and compare.

During our research we encountered a benchmark called MOT, that ranks multiple models every year. One of the most documented papers that has produced state-of-the-art results in recent years is FairMOT. Using their extremely detailed github page, we implemented the tracker.

Although their github page is very useful, the implementation was not as easy as we thought. We started implementing the model on a windows machine, using conda environment, just to find out that FairMOT can't run on a windows machine because a package called cython-bbox. From there we went on to use Oracle Virtual Machine Manager to create a Linux environment. Again, we used a conda environment on the Linux machine, the problem now was that we could not find a way to use the local Nvidia GPU on the virtual machine. Lastly, using google colab, we were able to implement and run the model with the google GPU.

We used a pre-trained model that was trained on pedestrians data. The input is an mp4 video and the output is an mp4 file with bounding boxes tracking players, and a text file presenting the results for each frame.The model ran ~13fps and produced average results. The results of the first model (Yolov3 + TensorFlow + DeepSort) were better, however, using the support and the detailed documentation FairMOT has, I believe that we can train it using soccer data, and make it a better option for our purpose.

Implementation:

Although their github page is very useful, the implementation was not as easy as we thought. We started implementing the model on a windows machine, using conda environment, just to find out that FairMOT can't run on a windows machine because a package called cython-bbox. From there we went on to use Oracle Virtual Machine Manager to create a Linux environment. Again, we used a conda environment on the Linux machine, the problem now was that we could not find a way to use the local Nvidia GPU on the virtual machine. Lastly, using google colab, we were able to implement and run the model with the google GPU.

Environment:

Google colab notebook. Google colab GPU.

Model:

We used a pre-trained model that was trained on pedestrians data. The input is an mp4 video and the output is an mp4 file with bounding boxes tracking players, and a text file presenting the results for each frame.

Efficiency and Results:

The model ran ~13fps and produced average results. The results of the first model (Yolov3 + TensorFlow + DeepSort) were better. The different errors are shown in figures X - Y.

Error types:

Players the are not detected (the player on the top left side of the field)

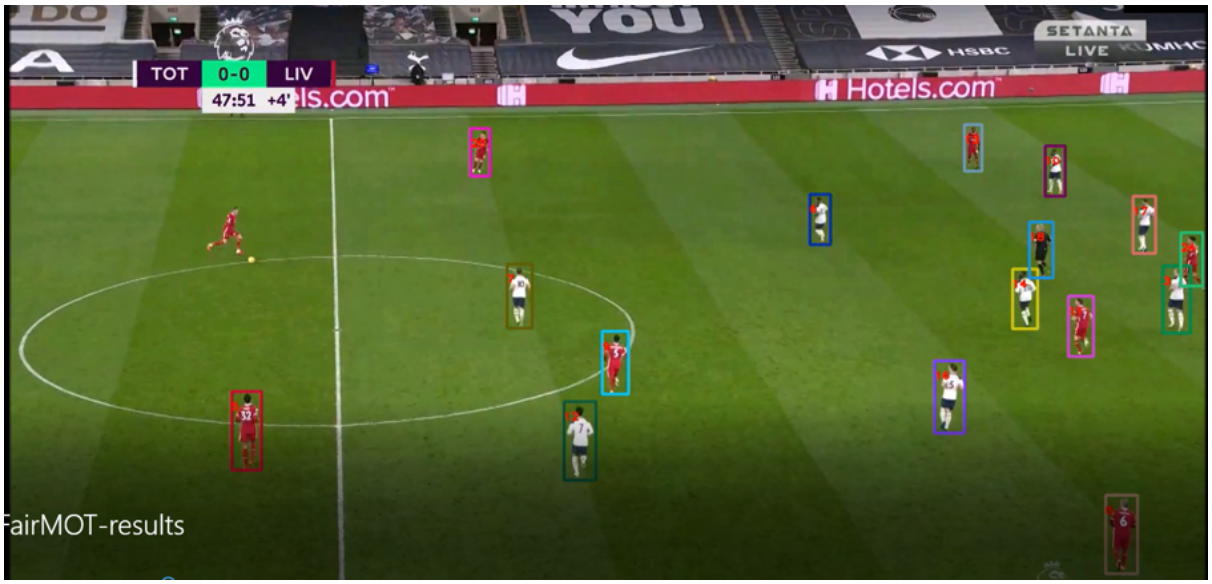


Figure 33: Some players did not get detected

Two players detected as one player when they are near each other, as seen in the image below, the two players in the middle of the field (detection number 9)

Figure 34: Error - one detection for two players

Future Use:

Considering the support and the detailed documentation FairMOT has, I believe that we can train it using soccer data, and make it a better option for our purpose.

6.7.5 Multiple Object Tracking - Methods Comparison

The end goal is to choose one of the tracking models that will best with the project workflow. In this section I will make a brief comparison between the two methods we implemented and try to recommend the better model. I will compare the two models based on a few subjects: efficiency, future use, results, environment, and implementation

Efficiency

The difference in speed between the two models was significant. The first model - Yolov3 + Tensorflow + DeepSort ran ~3fps while the other model - FairMOT ran ~13fps. It's a significant difference, however, I believe that with better hardware, the first model would run much faster and it would be usable. Bottom line, if I had to choose one of the two based on efficiency, FairMOT would be it.

Future Use

Looking ahead, we can split the future into near future and far future. In the near future, our goal is easy implementation and training options for our model. In the far future, our goal is a dynamic model that we can change on the fly when better methods are published.

There are two main differences between the models; Yolov3 + Tensorflow + DeepSort has a dynamic detector (Yolov3), in the future, whenever there is a better detection method, it would be very easy to replace Yolov3 and use a different detector, however, FairMOT implemented their own detector, and it would be difficult to replace it with another detection model.

Another major difference is training and documentation; both models give you the option to train with your own data, however, FairMOT has a very straightforward and organized documentation that can be of great help when we try to train, whereas Yolov3 + Tensorflow + DeepSort throw you into the deep water when it comes to training.

Bottom line, I would say that for the near future, FairMOT might be a better option, but for the far future and for the purpose of having a dynamic model, Yolov3 + Tensorflow + DeepSort is the winner.

Results

Eventually, one of the most important things is the final result that we provide to the user. We tested both models on the same short soccer video and the results were relatively good using either one, and they looked good when looking at them independent from each other. However, when comparing the results to one another there is a clear winner - Yolov3 + Tensorflow + DeepSort shows much better results when it comes to detect and track the players.

Having said that, both models participated in the MOT16 challenge and FairMOT displayed significantly better results, therefore, I think that when trained on the same relevant data, FairMOT has the potential to produce better results.

Bottom line, since we do not have available data for training, Yolov3 + Tensorflow + DeepSort is the winner in this category for now.

In the image below (coming from the official paper for FairMOT - “FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking”) you can see the result of FairMOT in the different MOT challenges. As you can see, in MOT16 both FairMOT and DeepSort participated, and FairMOT had much better scores.

TABLE 9
Comparison of the state-of-the-art methods under the "private detector" protocol. It is noteworthy that FPS considers both detection and association time. The one-shot trackers are labeled by "*".

Dataset	Tracker	MOTA↑	IDF1↑	MT↑	ML↓	IDs↓	FPS↑
MOT15	MDP_SubCNN [25]	47.5	55.7	30.0%	18.6%	628	<1.7
	CDA_DDAL [64]	51.3	54.1	36.3%	22.2%	544	<1.2
	EAMTT [65]	53.0	54.0	35.9%	19.6%	7538	<4.0
	AP_HWDPL [66]	53.0	52.2	29.1%	20.2%	708	6.7
	RAR15 [7]	56.5	61.3	45.1%	14.6%	**428**	<3.4
	TubeTK* [44]	58.4	53.1	39.3%	18.0%	854	5.8
	FairMOT (Ours)*	**60.6**	**64.7**	**47.6%**	**11.0%**	591	**30.5**
MOT16	EAMTT [65]	52.5	53.3	19.9%	34.9%	910	<5.5
	SORTwHPD16 [1]	59.8	53.8	25.4%	22.7%	1423	<8.6
	DeepSORT_2 [2]	61.4	62.2	32.8%	18.2%	781	<6.4
	RAR16wVGG [7]	63.0	63.8	39.9%	22.1%	**482**	<1.4
	VMaxx [67]	62.6	49.2	32.7%	21.1%	1389	<3.9
	TubeTK* [44]	64.0	59.4	33.5%	19.4%	1117	1.0
	JDE* [14]	64.4	55.8	35.4%	20.0%	1544	18.5
	TAP [6]	64.8	**73.5**	38.5%	21.6%	571	<8.0
	CNNMTT [5]	65.2	62.2	32.4%	21.3%	946	<5.3
	POI [4]	66.1	65.1	34.0%	20.8%	805	<5.0
	CTrackerV1* [68]	67.6	57.2	32.9%	23.1%	1897	6.8
	FairMOT (Ours)*	**74.9**	72.8	**44.7%**	**15.9%**	1074	**25.9**
MOT17	SST [69]	52.4	49.5	21.4%	30.7%	8431	<3.9
	TubeTK* [44]	63.0	58.6	31.2%	19.9%	4137	3.0
	CTrackerV1* [68]	66.6	57.4	32.2%	24.2%	5529	6.8
	CenterTrack* [70]	67.3	59.9	34.9%	24.8%	**2898**	22.0
	FairMOT (Ours)*	**73.7**	**72.3**	**43.2%**	**17.3%**	3303	**25.9**
MOT20	FairMOT (Ours)*	**61.8**	**67.3**	**68.8%**	**7.6%**	**5243**	**13.2**

Figure 35: Comparison between models

Environment

Setting up the environments for the models was maybe the most difficult part in the process, so I think it should be taken as variable when we choose the final model.

Starting with Yolov3 + Tensorflow + DeepSort, on paper, this should have been the more difficult model; not as much documentation, connecting different models together into one and dealing with different versions of each package. It was a little bit of a struggle, but after a few tries everything worked properly.

FairMOT was supposed to be easier, it's one model with a lot of documentation and support, but unfortunately it can't run on windows. Because of that, I had to try a lot of different environments and it took me a few days just to find the right machine to implement the model on. In reality, Yolov3 + Tensorflow + DeepSort is the much easier and more flexible option and is our winner in this category.

Implementation

After setting up the environments the next step was to implement the code. In this category we have a clear winner - FairMOT. In order to use YoloV3 + Tensorflow + DeepSort, we actually need to write code, and use logic to connect the three. On the other hand, FairMOT is like an api. After setting the environment, all we need to do is to call the main script with the video you want to track. Can’t be more simple than that.

Final Decision

It was difficult making the final decision, even though both models have some downsides, overall, both showed good results and had a lot of positives. We recommended going with FairMOT because of the ease of training, the support, the documentation, and the fps ratio. However, our sponsor thought YoloV3 + Tensorflow + DeepSort would be the better option because of his view on the future. He wants the model to be more robust with changes to the detector, and that model has a significant advantage when it comes to replacing the detector.

As a final decision, we went with the sponsor recommendation and chose Yolov3 + Tensorflow + DeepSort for the meantime.

6.8 Homography Estimation via Field Lines

6.8.1 Goal

The goal of this role is to receive input in the form of a video along with coordinates detailing the positions of targets that are being tracked. The input comes in two different formats: either the locations of specific targets throughout the entire length of the video regardless of whether they are on the screen or not - obtained through the online object tracking algorithm - or the positions of every current player in each frame via the player tracking algorithm. These outputs are then used as inputs into the homography estimation code to be used to determine player coordinates on a top-down view of a soccer field to find out the exact positions of every tracked player on the field.

There are two key requirements that are needed to compute the homography estimation then using that matrix to warp an image's coordinates so that we can find out the specific player coordinates on the soccer field; these requirements are semantic segmentation and camera parameter estimation. They will be each discussed in their relative sections below.

6.8.1.1 Resources

There were a number of immediate difficulties that became apparent when researching papers and models that can produce desirable results. The following are some of the problematic difficulties that were faced:

1. There is a significant lack of public models available for use (i.e. no code)
2. Homography estimation of field lines is an obscure topic with very little public information being shared due to papers keeping their findings private
3. Contains necessary dependencies/sub-models that are kept private
4. The greater majority of models that are capable of homography estimation are not related to sports
5. The models that can compute the homography estimation are incapable of being adapted for use and are purely for the purpose of the paper they stem from

6. Models contain neural networks with the training code privated so it cannot be adapted for the end goal of multiple sports support

Of the few models that do have publicly available code, only the following papers/project had code that are able to produce some type of usable results:

- [14] Optimizing Through Learned Errors for Accurate Sports Field Registration (Sportsfield Release)
- [15] Sports Camera Calibration via Synthetic Data (SCCvSD)
- [16] Hawk Eye

The Sportsfield release paper is able to compute the homography estimation from start to finish, the problem with it is that it only works with a single frame, is very resource heavy, and also very slow; however, the results are quite desirable and only the training code is unavailable. The following figure shows this paper's implementation of a warped image on a top-down view:

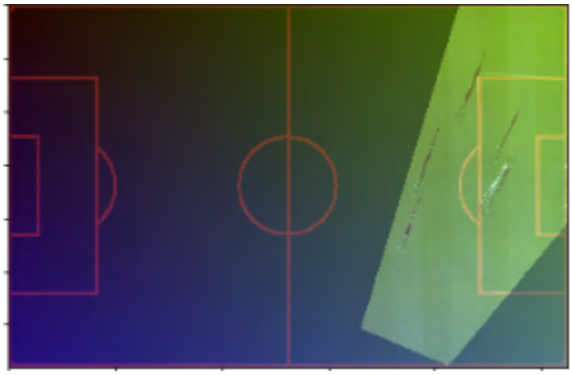


Figure 36: The Sportsfield Release implementation

The Sports Camera Calibration via Synthetic Data paper is able to again give desirable results similar to the Sportsfield release code, however, it requires the use of a second model for input that first is used to create Generative Adversarial Networks (GANs) to extract the relevant field lines and remove unnecessary information. The following

figures show the GAN necessary as input before being processed into the top-down view:

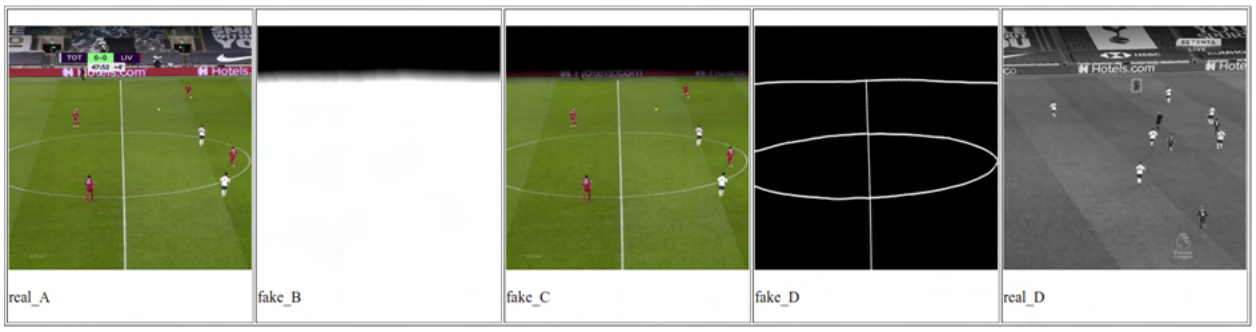


Figure 37: GAN for SCCvSD

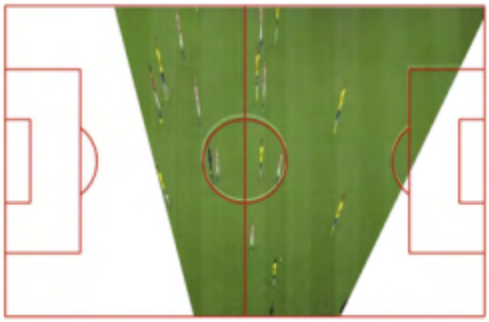


Figure 38: The SCCvSD implementation

Finally, we have the Hawk Eye project. Like the previous two papers, this model is able to, again, successfully create a homography estimation that warps each point in image onto a top-down view of a soccer field template; however, this model requires previously known homographies of the input images and also contains a vast amount of hard coded variables that are not feasible to have in real world applications. The following figure shows the Hawk Eye implementation of creating a top-down view of a frame from a soccer broadcast video:

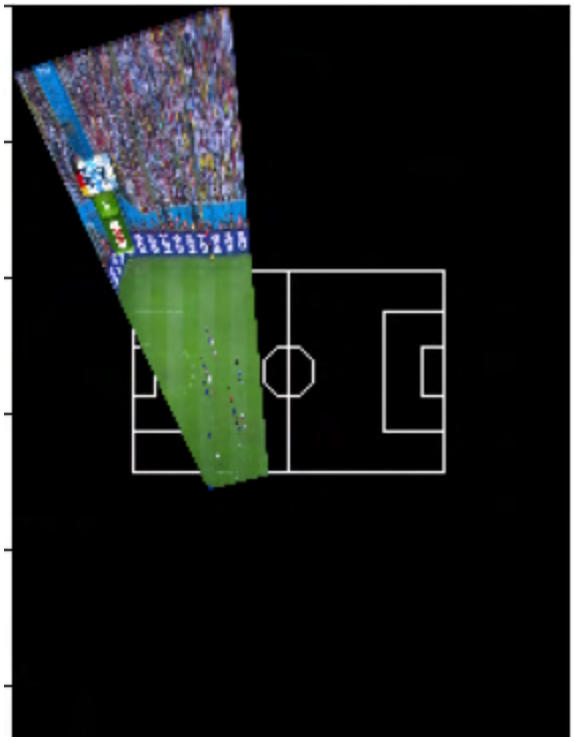


Figure 39: The Hawk-Eye implementation

From the current options, the best model to use is from the code provided by the Optimizing Through Learned Errors for Accurate Sports Field Registration paper. With small modifications to code and adding support for the various inputs received from the other algorithms used in this project, this model will give us a way to test the entire project pipeline while we work on implementing our own way to achieve the goal that supports every requirement and/or stretch goal listed.

6.8.2 Semantic Segmentation

Semantic segmentation is the task of labeling every pixel in an image with what the used model has been trained on. The semantic segmentation model will take in an image and then, depending on the value of every pixel in that image, it will attempt to classify and label every pixel. Using the [17] DeepLabv3 semantic segmentation model trained specifically on labeling only bears, it will output a result where the bear is the only pixel that is different; as shown in the figures below:

Figure 40: Image of a bear before semantic segmentation is applied

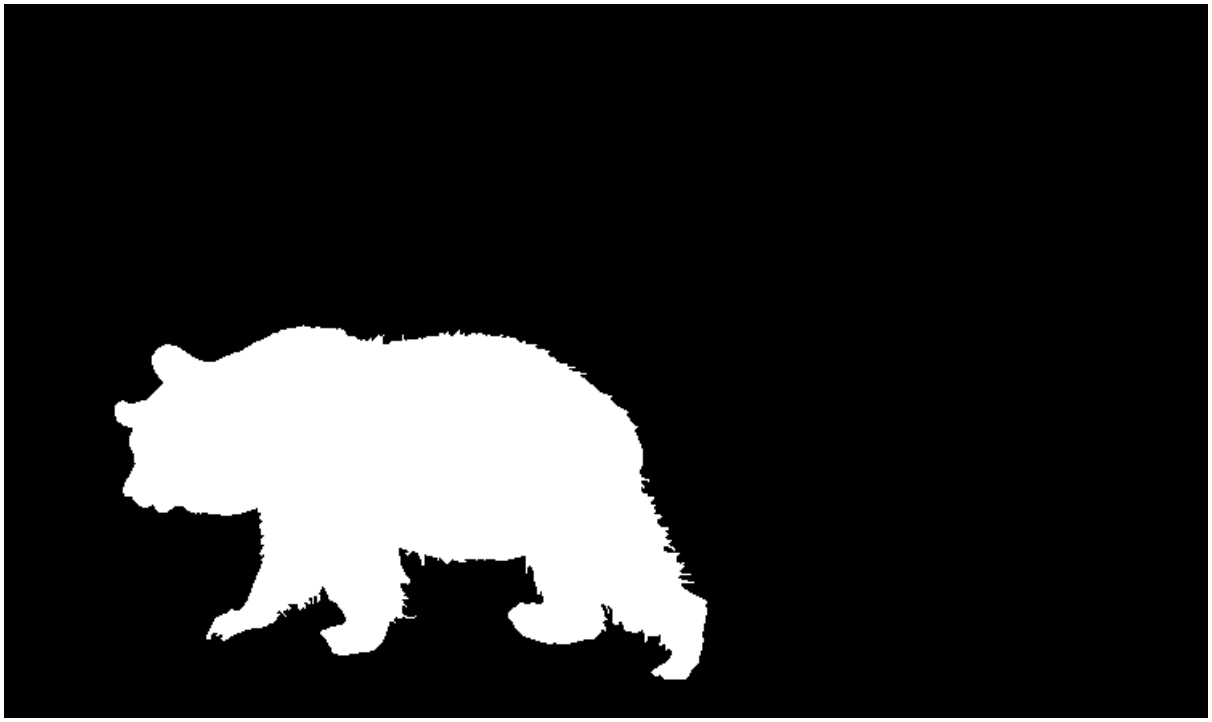


Figure 41: Image of a bear after semantic segmentation is applied

The point of using semantic segmentation in this project is to be able to get rid of and label everything that is not a field line in a given image or to otherwise specifically label the lines via known pixel color. This is so that the homography can be done and the given image of a soccer field can be fit onto a top down view of the field. As such, the goal of this task is to successfully be able to take in a frame and output the segmented version.

6.8.2.1 Implementation of Semantic Segmentation

There are a number of different pre-trained models that exist for different frameworks. However, there are almost no pre-trained models that exist for semantic segmentation of soccer field lines and virtually no datasets that can be used to achieve satisfactory results. While the models are plenty, they are all trained on datasets that consist of cats, dogs, humans, cars, buildings, and the like. As such, we will be attempting to figure out how to train our own model on a significantly limited custom dataset or try to understand and piece together a multitude of different methods all with their own drawbacks and to then optimize as well as make it significantly more efficient and in line with the goal of this project.

After a large amount of struggling to find models and testing different models such as Deeplab and PixelLib, we decided on using the code provided from the paper Optimizing Through Learned Errors for Accurate Sports Field Registration (Sportsfield Release) for the initial implementation and so that we have some results to show and build off of for the other roles in the project. It will be switched out due to the high resource allocations and time needed to produce results as well as the incompatibility with the final goals of this project. The first implementation of the semantic segmentation resulted in satisfactory results provided from the model. Using the figure below as input, the segmented field lines are shown as follows:

Figure 42: Input image of soccer field before semantic segmentation

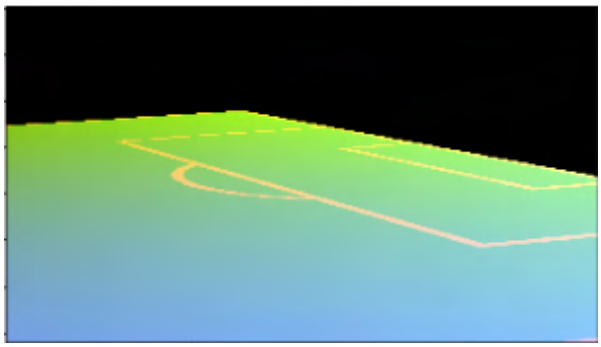


Figure 43: Output image of soccer field after semantic segmentation

Thus giving us the semantic segmentation of the field lines and accomplishing the goal of the semantic segmentation role. This frame can then be fed into the role of camera parameter estimation for computing the homography.

6.8.3 Camera Parameter Estimation

Camera parameter estimation deals with determining the estimated homography of a given image to transform the perspective in some way. In this project, the homography is the isomorphic projection of a frame taken from a broadcast soccer video onto a template of an entire soccer field from a top down view of which the frame exists in that view. The overview of how this works is to do semantic segmentation of the frame to get the field lines where it will then be transformed and warped to fit how a top down view would look; afterwards, using the warped view, the homography is determined based on the lines in the frame with the lines in the top down view. A visualization of imaging warping through a homography matrix transformation is shown below.

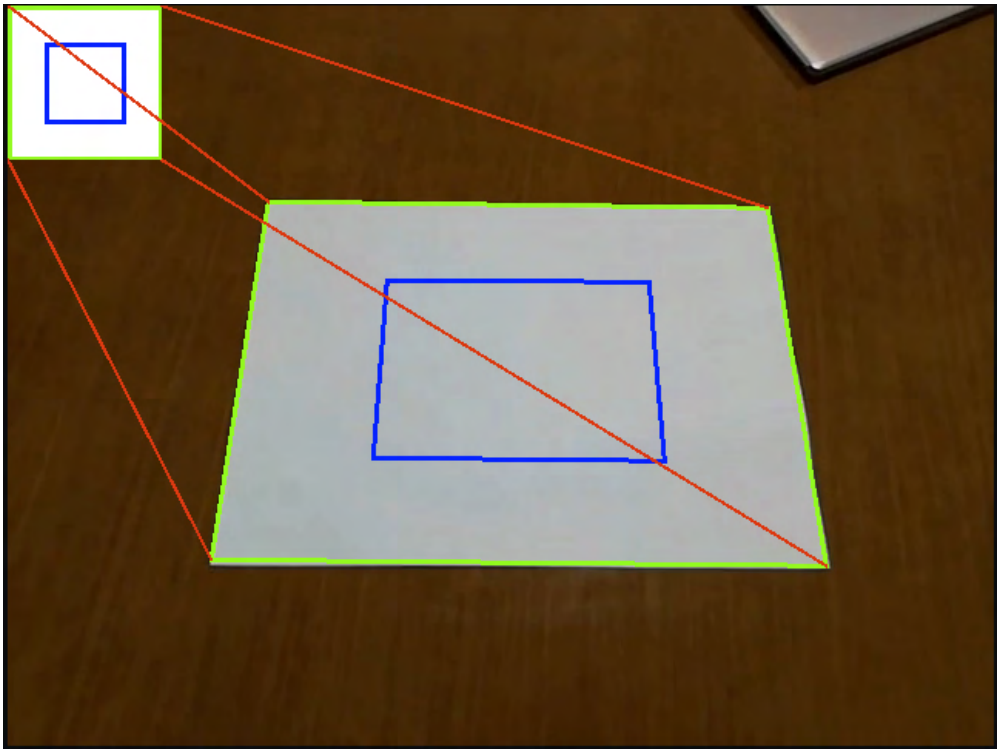


Figure 44: Visual representation of image warping through usage of homography matrices

6.8.3.1 Implementation of Camera Parameter Estimation

The Sportsfield registration model being used is made up of two networks based on resnet18 architectures. The first one, the initial registration network, is used to provide a rough estimate of the sportsfield registration, parameterized by a homography transformation. The second one, the registration error network, takes the output from the first network and optimizes the initial homography estimation using the gradients provided by differentiating the images and through error based learning. The goal is to bring the loss between the field lines in the warped image and on the template image to a minimum. This second network will run for as many iterations as provided or until a loss of 0 is achieved.

The actual runthrough is as follows: Create the initial registration network and registration error network and load the pretrained weights into them. Then, generate the initial homography estimation. Take the homography generated in the previous step and use it to warp the frame onto the field template for input into the second network. Use the second network to minimize the least absolute deviations (L1 loss) between the field lines in the warped frame and the soccer field template for a number of defined iterations. Once the number of optimization iterations has completed, find the argument of the minimum of the obtained losses (argmin) for use as the optimal value for the homography matrix.

The specifics of the homography estimation and optimization is shown below:

- Initialize the pretrained models
    - Initialize user options
    - Determine the criterias
    - Build the initial registration network
        - Load a resnet18 model with pretrained weights for initial homography estimation
    - Build the registration error network
        - Load a resnet18 with pretrained weights for surface regression
- Generating initial homography

- Determine lower corner points (3/5ths) of a normalized image with the width and height set to 1 using predefined default corner points: [(-0.5, 0.1), (-0.5, 0.5), (0.5, 0.5), (0.5, 0.1)]
- Determine the corners of the input frame, reshape and permute to get another set of coordinates
- Using direct linear transformations between the two sets of coordinates, we obtain the initial homography

- Warping image
  - Take the corner coordinates from the previous step and append them to a list
  - Using the initial homography, warp the frame onto the field template
    - Create a tensor meshgrid from of size [-0.5, 0.5] for x and y coordinates each
    - Perform a flattening operation on x and y
    - Stack (bind) x and y then append ones in z index the for homogenous coordinates: [x, y, z]
    - Do matrix multiplication with the initial homography and the stacked xy coordinates to get the warped coordinates
    - Split the z index off of the xy coordinates
    - Unbind the x and y coordinates from each other
    - Create a grid from the new warped coordinates
    - Compute the warped image from the initial image and the flow-field grid using the values from the image with the locations of those values via the grid

- Obtaining the loss between the warped image and the field template
    - Pass the warped image into the second network which will find the gradient of the image with the homography then using this, it determines the estimated distance between the lines
    - Get the L1 loss using the estimated distance
    - Append the loss
- Finding the optimal homography
    - Get the corners of the image that had the smallest loss value
    - Using the corners with the reference corners of:
      [-0.5, 0.1], [-0.5, 0.5], [0.5, 0.5], [0.5, 0.1], use direct linear transformations to obtain the optimal homography

To visualize this process, we use the following frame as input along with the desired field template:

Figure 45: Input image of soccer field before homography estimation

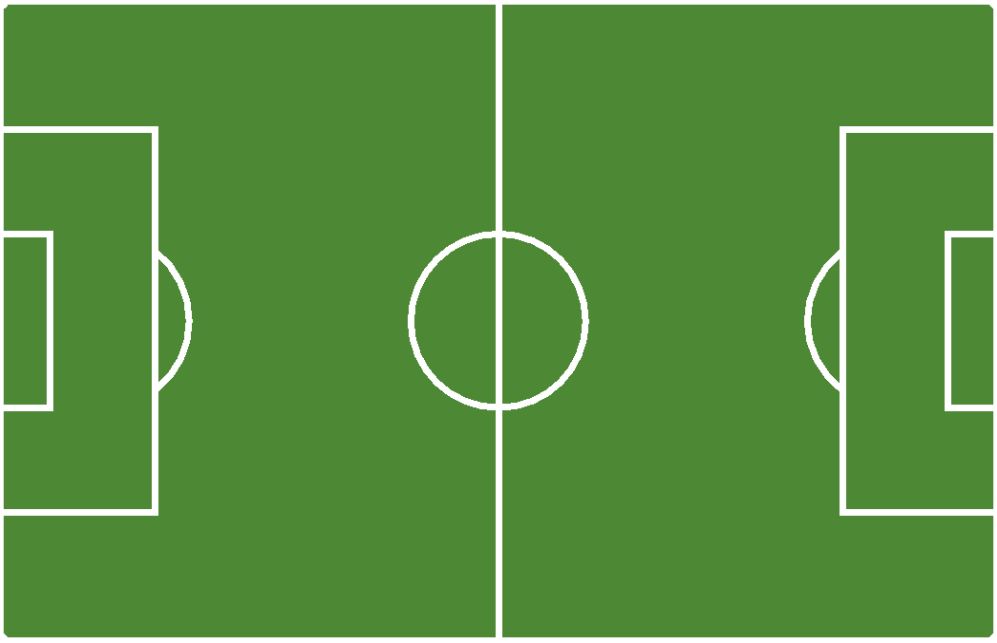


Figure 46: Template image of top-down view of soccer field

The outcome after the model is run will give an estimated warped image like below. This is the warped image of the initial estimated homography before optimization.

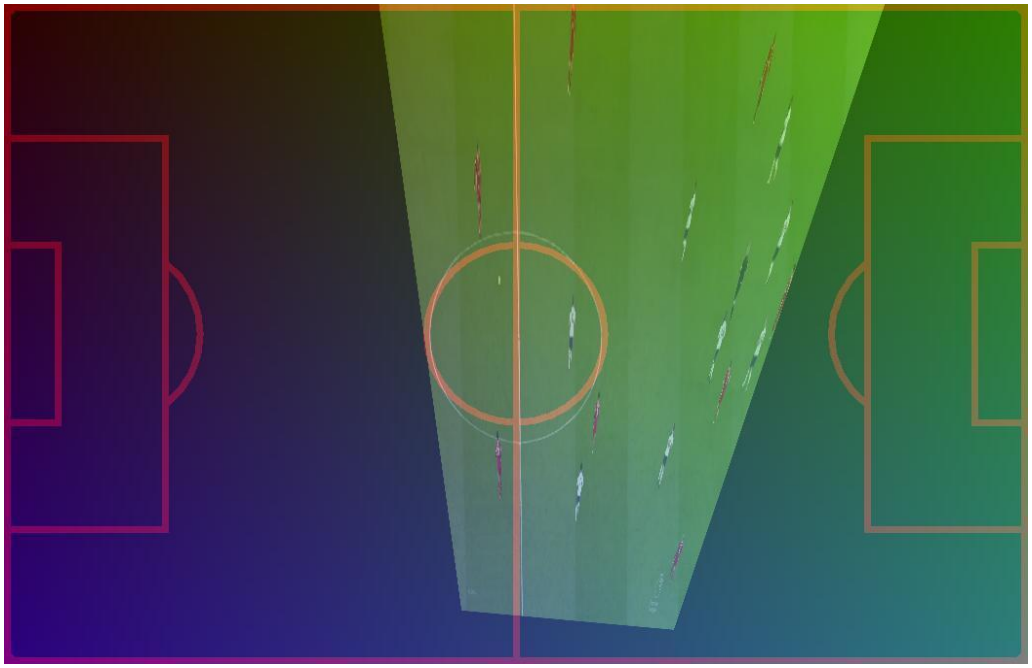


Figure 47: Warped view of soccer field onto template image after homography estimation and image warping is applied

This gives us the specific frame fit onto a top down view of the field for the purpose of being able to note down every players' positions. The bottommost region of the frame is where the camera originates and, as the camera pans or zooms in, it will also move in order to give a view that is as accurate as possible.

6.8.4 Determining Player Coordinates

After finding the homography of the frame based on the top down view, the next step is using positional coordinates so that exact player locations can be shown. There is two different possible inputs used to determine where in a frame a player is located. The first possibility is via the real time player tracking algorithm. This will give the location of one or more players that need to be tracked. The other input is from the player tracking algorithm which will supply the coordinates of a frame of every single player on the field.

After the estimated homography is determined and where the frame lies on a top down view is shown, the same transformations applied to the initial frame must also be applied to the coordinates. As such, when given the x,y, width, height coordinates of (966, 101, 23, 34), we get the following frame:

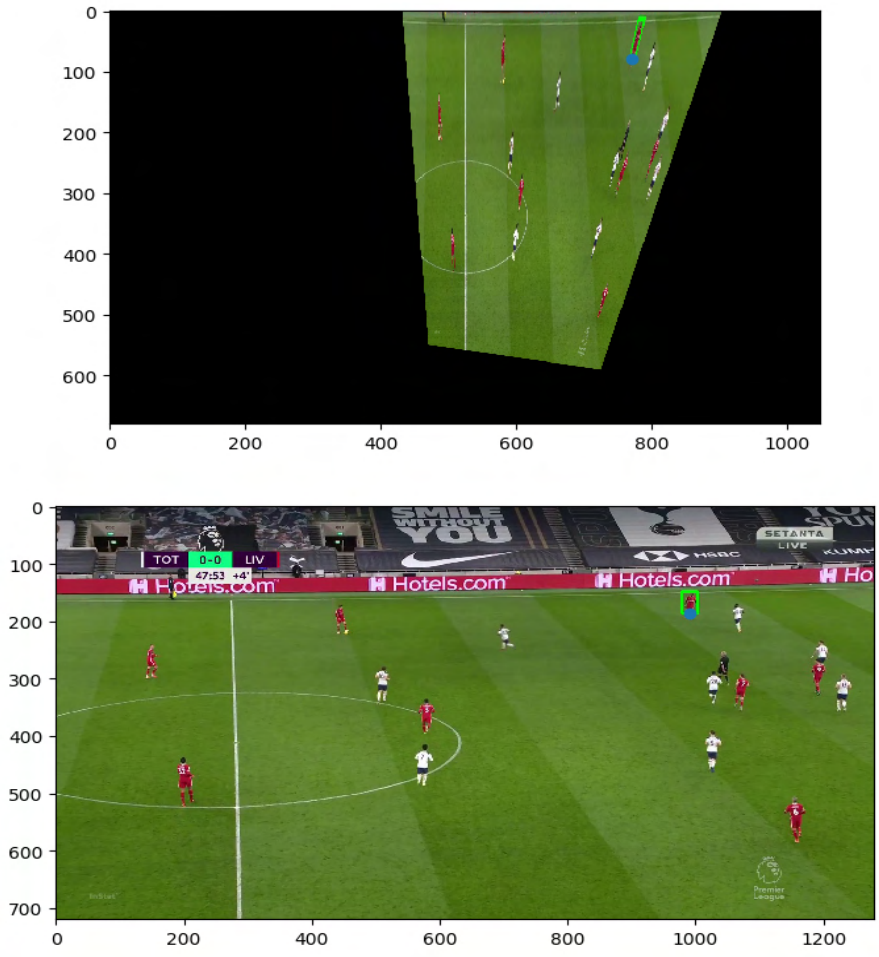


Figure 48: Input frame of soccer field from the video feed including marked player (bottom image) which is then warped (top image)

The green box represents the tracked player, and the blue dot is the position of the player's feet. This can then be done in a similar fashion with however many players are required to be tracked on the field. The following set of images represent different positions on the field.

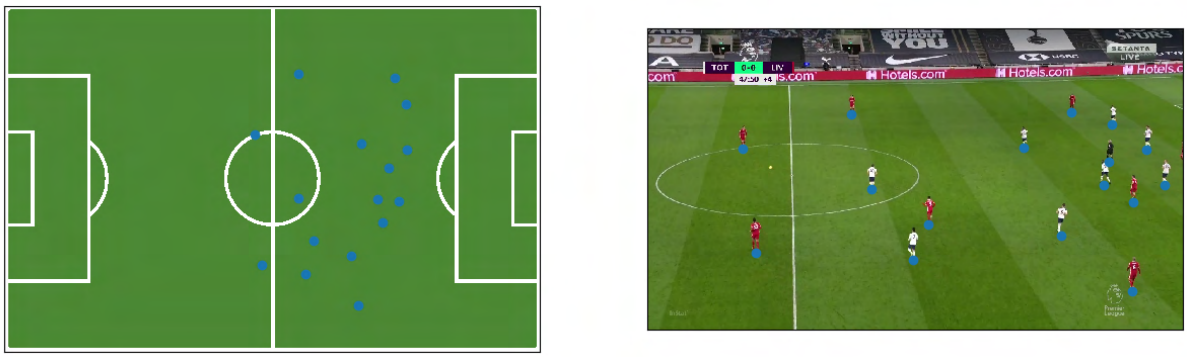


Figure 49: Center field input and output before and after homography estimation

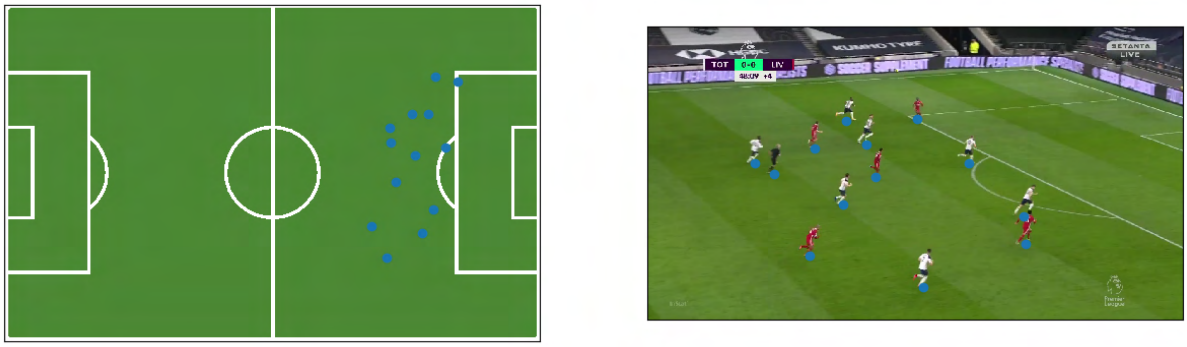


Figure 50: Right field input and output before and after homography estimation

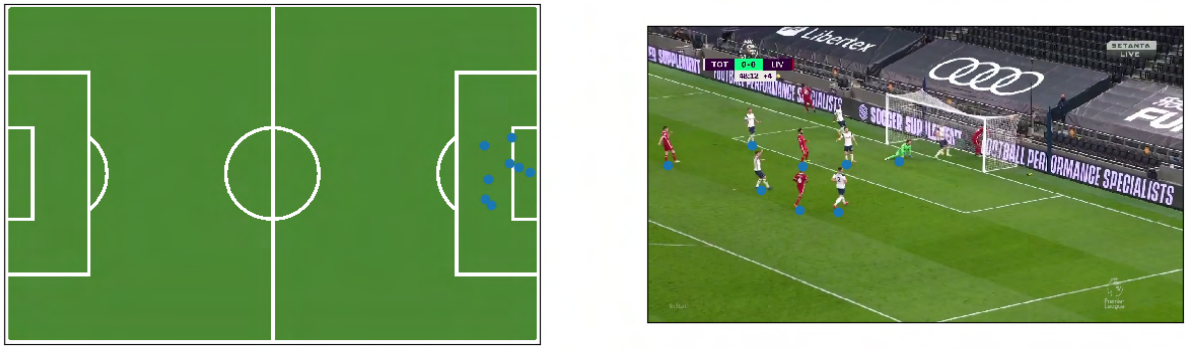


Figure 51: Right goal input and output before and after homography estimation

The homography estimation works by taking the frame and comparing it to a list of camera angles and known homographies. Then, it will use the line segmented image concatenated with a field template, then through error based learning and back propagation, will tweak the weights of the homography through a given number of times desired. The output of this is a homography matrix.

In order to determine the exact coordinate of the player on the warped view, we first use the input given from the tracking algorithms. These inputs are in the form of (x, y, width, height). Since the input given from the tracking algorithms will be a box from the top of the players' heads to their feet, therefore, to get the feet placement $\kappa\nearrow_{\mathbb{I}}\bar{o}\searrow_{\mathbb{I}}\lambda$, we can do:

$$\nearrow_{\mathbb{I}\,\epsilon} \quad \nearrow_{\tau} \quad \frac{\nwarrow \nearrow \Pi \Downarrow \lrcorner}{\acute{\alpha}} \bar{o}$$

$$\searrow_{\mathbb{I}\,\epsilon} \quad \searrow_{\tau} \quad \lrcorner \sum \Pi \lrcorner \downarrow$$

In similar fashion to how the warped image gets created, to find the feet coordinates of the warped image, we can use matrix multiplication of the estimated homography of the image and $\kappa\nearrow_{\mathbb{I}}\bar{o}\searrow_{\mathbb{I}}\lambda\,\emptyset$

6.8.5 Difficulties and Improvements of the Methods

6.8.5.1 Difficulties

The biggest problem with using two networks to determine the optimized homography lies with how fast and accurate the algorithm is. With speed comes a cost of accuracy which is not always the best choice of action. With the current implementations, it takes between 3 and 10 seconds per frame in order to output relatively desirable results. Otherwise, if the number of iterations done per frame is lowered, the speed increases, but the accuracy is much lower. In the pursuit of trying to find a balance, the seemingly optimal number of iterations for an initial run is 80.

A visualization of the results is as follows:

Using a low number of iterations over the frame (20 at 1 second per frame):

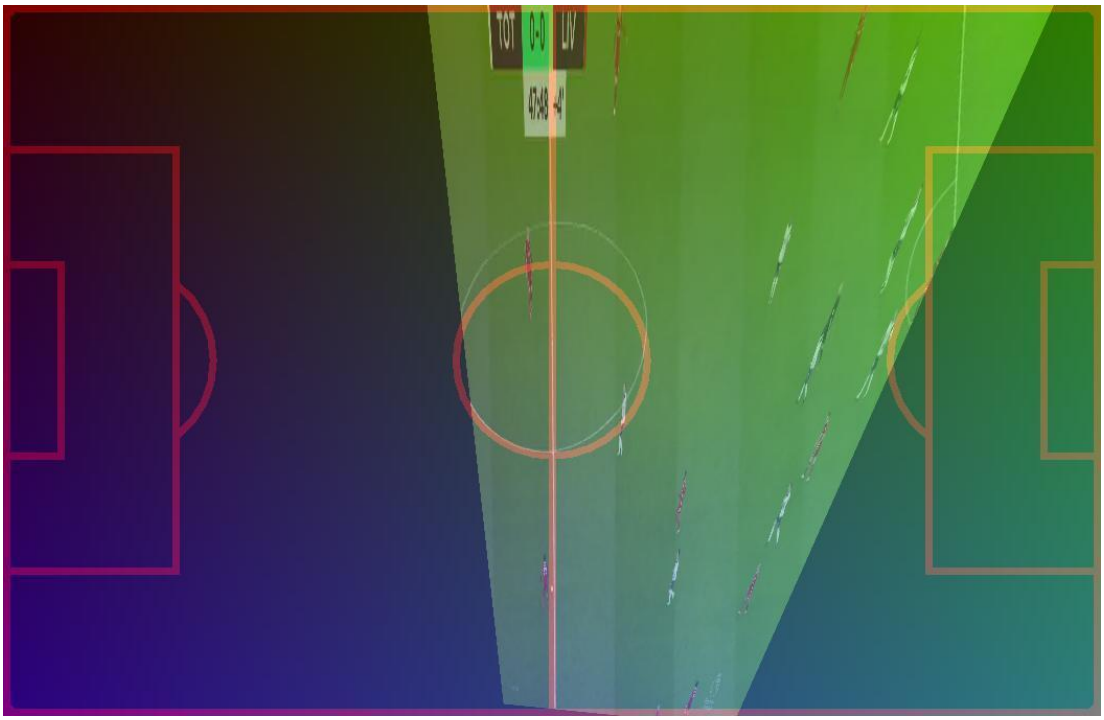


Figure 52: Warped image after homography estimation using 20 iterations

Using a higher number of iterations over the frame (80 at 4 seconds per frame):

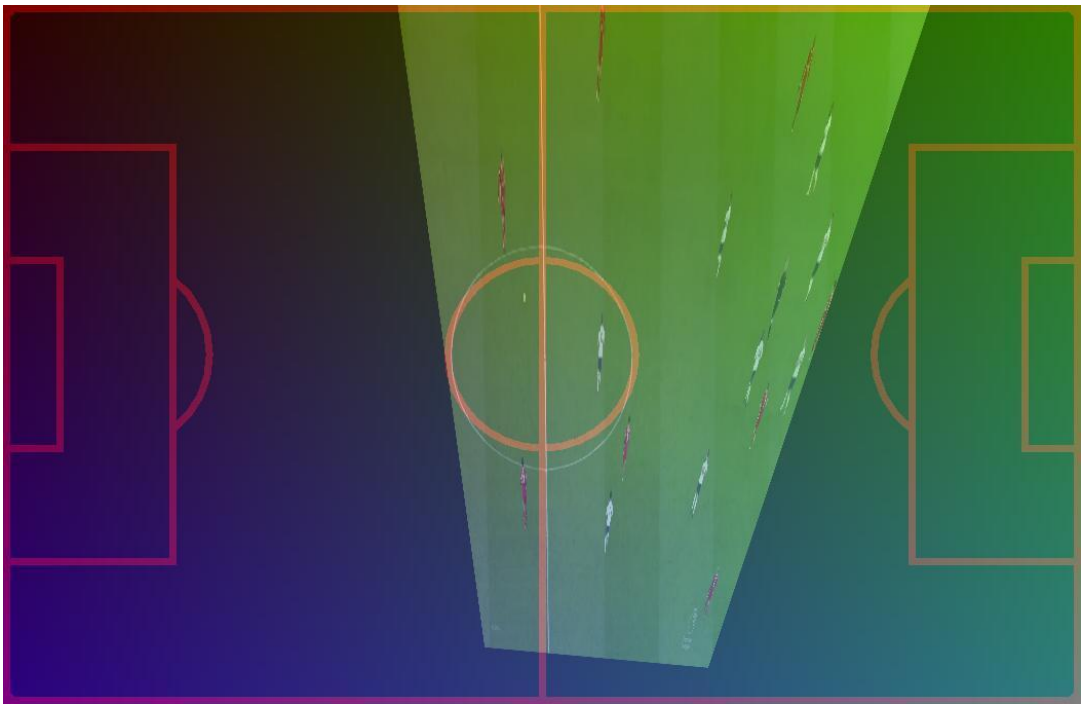


Figure 53: Warped image after homography estimation using 80 iterations

The reason there is such a significant change in where the frame is being warped is due to the model using error based learning and back propagation in order to optimize where the field line is actually occurring. The model uses the field lines and changes the weights in order to try and minimize the error between the estimated location and the actual location of where the line is. The major problem is that for a small improvement in the accuracy, there is a 400% increase in computational time.

This problem also relates to the accuracy of the entire algorithm. Attempting to balance out speed and accuracy in the previous steps results in losses in future steps as well. If the homography is inaccurate, then displaying player coordinates will also be inaccurate as they are based off of the aforementioned.

To account for the accuracy and speed problems, we will be implementing a multi-stage system where the first run through the camera parameter estimation model will be using the optimal number of iterations while accounting for speed and accuracy as previously stated (80), then it will do another run with a significantly higher number of iterations that can be ran for a longer time. As the point of this project is to create an interactive web app, using the first state of this model will allow the user to see the initial results which will then be improved and updated.

The next problem lies with memory management. As it was, the algorithm loads the frame, the homography, and everything else required into memory. As this is being done on a video and the algorithms come from models which work on images alone, when modifying everything to work together, a 30 second clip loads almost 40Gb worth of variables into memory. Along with python's inherent disadvantage of not having friendly memory management tools, we had to come up with a few workarounds.

Python processes will not always release memory that it uses back to the operating system so we will have to force this to happen. In order to do that, the main constituent of the memory problem - creating the homographies for every frame in the video - gets forked on every frame so that it works in a child process. After the child finishes and returns the desired results, the child process gets deleted and thus any memory that was taken by it was returned to the operating system as desired.

The next problem was that the models had no way to save and load the generated homographies for future usage. This is important as being able to save and load the homographies will mean they only need to be calculated a single time instead of any time the model is called. The solution to this was the built-in PyTorch save method which is called on a per frame basis with a flag denoting whether it is loading homographies or generating them. Now, whenever this model is called, it will check if homographies already exist where it will then load them from the database, otherwise, it will be uploaded if not.

6.8.5.2 Improvements

There are many things that need to be tweaked for this model and overall role before they are finalized. Some of the improvements include better utilizing resources as well as adjustments to parameters and methods to increase speed/accuracy.

Future changes made will include many/all of the following:

- Determining the optimal resolution of images that will allow for less resource usage and increase speed, but not jeopardize accuracy
- Necessary number of frames to get desired results (15fps vs 60fps). Not all frames are required to get the player coordinates and will increase speed
- Creating child processes to force python programs to return memory back to the operating system

- Multithreading to increase CPU/GPU computational ability
- Constant garbage collection for freeing memory
- Determining where the optimal balance between accuracy and speed is

In addition to the above, we will be moving towards creating a custom semantic segmentation and homography estimation model. This is due to the limitations set with using models from research papers. These models are not necessarily designed for speed nor are they suitable for real world application, they are made to prove a theory and/or increase accuracy in the best of conditions. For the implementation required in this project, a more suitable model would be one that is built for this specific purpose. In this case, using a custom built model designed for homography estimation in a real world application - where speed and accuracy both play very important roles - will be the best plan of action going forward.

6.9 Online Object Tracking

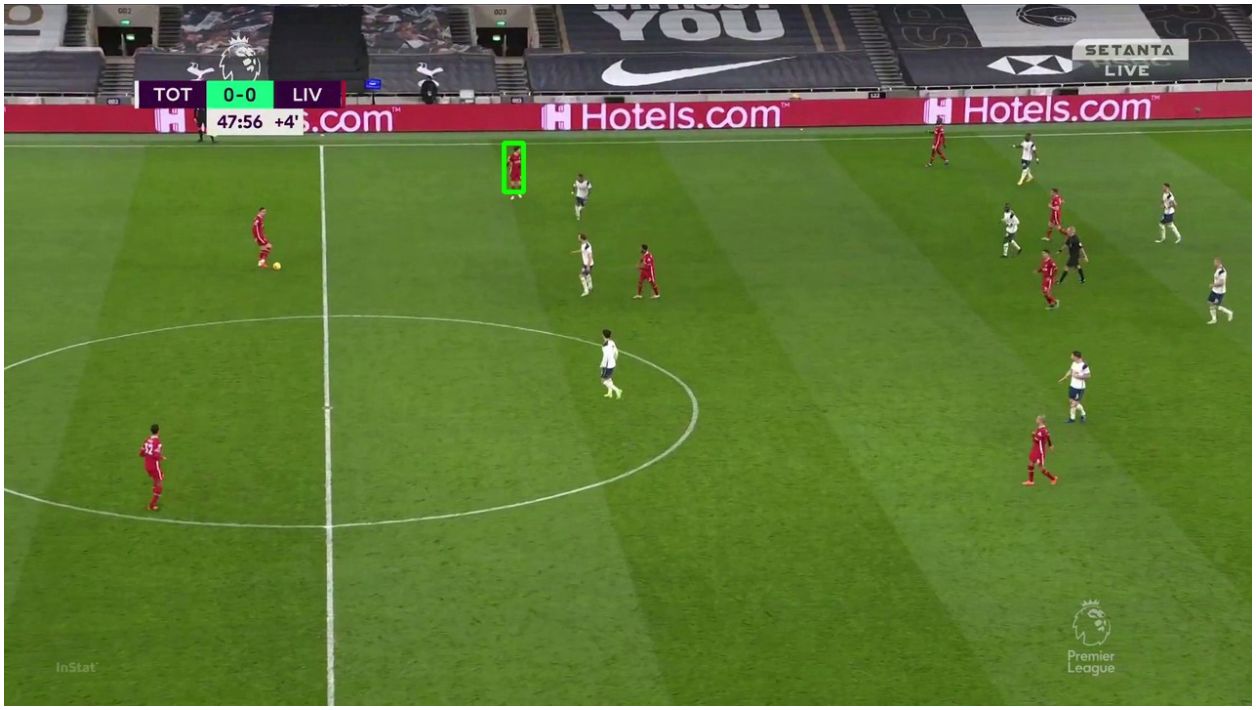
6.9.1 Fundamental Overview

Online object tracking, otherwise known as real-time object tracking, is the name of a type tracking algorithm designed with three primary differences from other forms of object tracking:

1. There is some level of human input involved to initialize the program and determine the starting parameters.
2. The algorithm runs in real-time without any future knowledge of what the next frame will be.
3. This algorithm will attempt to track only one object for the duration of its runtime.

These aspects have multiple consequences on design. In order to run an online object tracker the program has to have a static frame for the user to annotate which in turn indicates the region that contains the object to be tracked. This annotation most commonly comes in the form of a “bounding box”. A bounding box is drawn on the screen by the user by dragging their cursor from one corner of the area they want to select to the opposite corner. This creates a rectangle on screen which contains the object the user wants to track. If the area is incorrect the user can simply draw a new bounding box (which deletes the old one) and then confirm their selection. Once the user confirms their selection the algorithm starts to run and attempts to track the bounded object by comparing the difference between frames and moving and resizing the bounding box accordingly to keep the object inside it. The bounding box itself is represented by four values: an x and y-coordinate representing the location of one of the corners and the width and height of the box representing its size.

As mentioned above, the user can only draw one bounding box per run of the algorithm, and attempting to draw a second would delete the first selection. This means that the visual feedback from running the algorithm should only display one bounding box on the screen at any one time, as displayed below.



The other primary consequence on design is that real-time object trackers have to be lightweight and fast. A highly complicated and accurate real-time tracker is useless if it cannot actually track in real-time. This presents a huge challenge for these algorithms because they have a maximum amount of time they can take on each frame which is also dependent on the framerate of the video. Therefore, while accuracy is highly valued we also want to make sure we prioritize choosing a real-time algorithm that keeps its functionality simple and works fast.

6.9.2 Relevance to Other Project Aspects

The use of this algorithm is intended to allow the user to manually create new tracks over parts of the video where there were false negatives, or in other words, missed detections. The user can run this algorithm from the first frame of missed detections to the last frame to create a new detection track. This new track can then be merged with the primary track for the player to add the new otherwise missed detection and data. Ultimately we want zero missed detections so we have perfect data and this helps reduce or even eliminate how often a player appears on the screen without being detected.

In its base form most of these algorithms don’t save any of the data or modified frames, and if they did they weren’t in the proper form or formatting. Luckily all of the relevant data (x-coordinate, y-coordinate, width and height) is available to gather each from the

program for each frame. This is then stored in a text file which can be parsed and understood by the camera parameter estimator to transform the data into a more useful form relative to the actual soccer field.

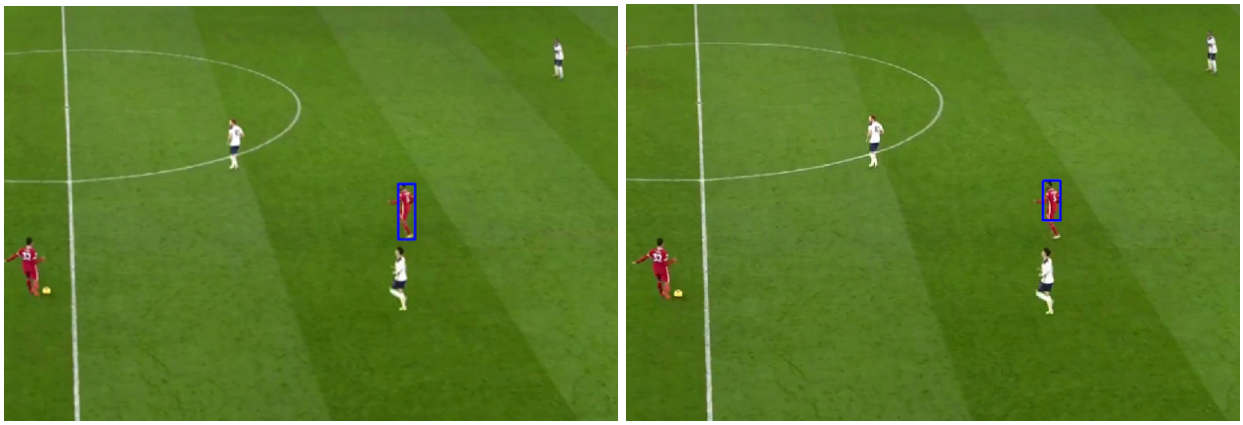
6.9.3 Potential Algorithms

When searching for a potential algorithm to use we want to take into account all of the earlier talked about points but there are also some other points to consider.

1. Accuracy of the tracking when two players cross paths.
2. Necessary level of user precision when drawing the bounding box.
3. Additional features that could be useful, useless, or even detrimental.

The first and arguably most important point of those listed above is the tracker's accuracy. This applies in a multitude of situations, however the most common and relevant iteration of this comes in the form of two players coming into close contact or even overlapping. Adding more players to the mix of course exacerbates the issue even further. No tracking algorithm I tested had perfect results and some were better in certain situations while faltering in others. Overall consistency was certainly an issue and it all stemmed back to the primary source of variance in the system: user input.

Depending on how the user draws the bounding box the output can vary quite dramatically. Small changes on a pixel by pixel scale can matter, but here we're more interested in the general shape and location of the bounding box. For instance, the bounding box can either be drawn to cover the entire player or only partially.



SiamMask + SiamMaskE

SiamMask and it's newer, modified version, SiamMaskE, are well known and powerful algorithms for real-time object tracking. They are both highly accurate and utilize a convolutional siamese tracking network.

The primary issue with these algorithms is that they're actually a bit more feature heavy and complex than we would like for what we're trying to accomplish. Both feature rotating bounding boxes and partial semantic segmentation.They implement these in part to try and push the bounds of what real-time tracking is capable of. Currently a real-time tracker with just full time semantic segmentation is nearly computationally impossible, but SiamMask attempts to bridge the gap between traditional bounding boxes and semantic segmentation by primarily tracking with the bounding box, but also applying the visual segmentation to the tracked object. In addition, the rotating bounding box could potentially present problems and inconsistencies for the camera parameter estimation.

The secondary issue is that these extra features add a lot of extra clutter to the screen and can be excessive and confusing for what we're trying to accomplish. The user doesn't need to see the semantic segmentation for this algorithm especially because if the tracking goes awry then the segmentation looks incredibly messy and unprofessional.

All in all, the extra features are unnecessary both visually and in terms of the data we are trying to gather from real-time object tracking.

Pysot

Pysot is a software system and research code base that uses a PyTorch deep learning framework that implements a variety of pretrained tracking algorithms and models. This includes SiamMask, SiamRPN++, DaSiamRPN, SiamRPN, and SiamFC which could use the following backbone network architectures: ResNet, MobileNetV2 and AlexNet. Here we will take a look at some of the other versions of SiamMask and, more prominently, SiamRPN++.

SiamMask_r50_l3

SiamMask_r50_l3 is the primary version of SiamMask included in the Pysot code base and uses the ResNet backbone in its implementation, however it has significant shortcomings in all of the same ways that SiamMask does due to SiamMask being a derivative of this algorithm. In addition the extra features discussed in SiamMask's

primary entry weigh down the algorithm's potential speed and efficiency, only able to hit speeds of up to 56 frames per second.

SiamRPN_alex_dwxcorr

This is a SiamRPN++ algorithm using an AlexNet backbone. It is a very simple tracking algorithm without rotating bounding boxes or extra, unnecessary features. It's very efficient to run by comparison to other algorithms in the Pysot code base. It boasts a massive speed of 180 frames per second, which is more than double that of any other algorithm. In addition, in testing, it was found to be one of the most reliable algorithms in terms of consistency between runs.

SiamRPN_alex_dwxcorr_otb

This version of SiamRPN is fundamentally similar to SiamRPN_alex_dwxcorr with only a few key differences. The OTB at the end signifies that this version of the algorithm was designed and trained for the OTB long-term tracking challenge. In testing it was found that this version was even more reliable than the standard version. Players getting close to each other or overlapping were far less likely to mess with the tracking and in general the drawing of the bounding box didn't need to be quite as precise, whereas that was still a significant issue for other real-time tracking algorithms.

6.9.4 Why SiamRPN_alex_dwxcorr_otb

Ultimately the SiamRPN_alex_dwxcorr_otb algorithm was chosen for the task of real-time object detection. This was decided on due to its innate reliability, speed and because it otherwise best met the criteria for a potential algorithm.

7 Overall System Design

7.1 Web Application Dashboard

The web application front-end will consist of Plotly Dashboards. Dashboards are deployed on a server like web applications, not like static HTML files. We will be mainly using Dash Components (dash core components, dash bootstraps components). We can find different visualization libraries like Matplotlib (which generates static images), Seaborn (completely integrated to the pandas data frame) and Plotly(which allows interaction with the plot like zooming, hovering and selecting).

Dash is the library used to create these interactive dashboards natively in Python. They offer the possibility of interaction between the different dashboard elements to have dynamic behavior. We will also be using Plotly graphs as a component to the dashboards.

The Web Application will consist of a dashboard dedicated to video trimming, to edit the footage during preprocessing(admin side), an annotation dashboard to assign and modify tracks(admin side) and a web application containing a search engine to query the already processed video database (user side). Each dashboard will have a basic navbar, as a common dash bootstraps component, that will allow each kind of user to navigate to the web apps home and other accessible dashboards.

7.2 Video Trimming Dashboard

Once the outside users(coaches, managers and technical staff) submit the sports video for pre-processing, the inside users (the Sports Science Teams) will be able to log in the web portal on the admin side and edit the video before it undergoes processing.

We are implementing a dashboard that will allow video trimming. This will consist of a UI that will allow the inside user to delete a series of selected frames to crop any footage that does not need to be analyzed. The user will be able select a range of frames using the interval slider. The dashboard also includes video playing functionality, and components that will allow you to skip ahead in the footage and rewind. See the image below to see the components and current implementation of the video trimming dashboard.

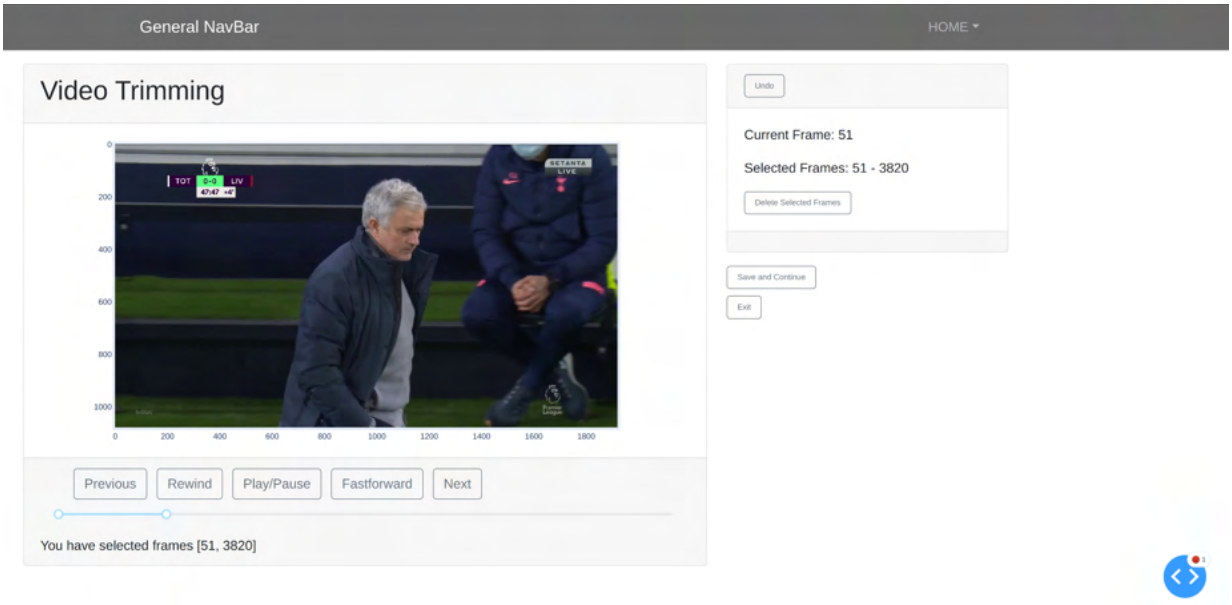


Figure 54: Video Trimming Dashboard Implementation

Track Annotation Dashboard:

This dashboard will be part of the post-processing view. The processed video frames will be available to the admin user for track annotation. The dashboard includes:

- A *Tracks* card that toggles to show a list of all tracks on the video frames, a list of the viewable tracks (the tracks on the current frame) and the tracks that have been assigned to a specific player. Each item on the list expands into the selected track's information.
- A *Players* card that toggles to show a list of the players for each team (A and B), and each item on the list expands into the selected player's information.
- A *Video* card, where the user will be able to play the video, change into the previous or next frame, as well as go through a whole section using the slider. The user will also be able to toggle into the different views to either hide or show all the overlays for the tracks in the whole video, and either hide or show all the overlays for the tracks in the current frame.

See the image below to see the components and current implementation of the video track annotation dashboard.

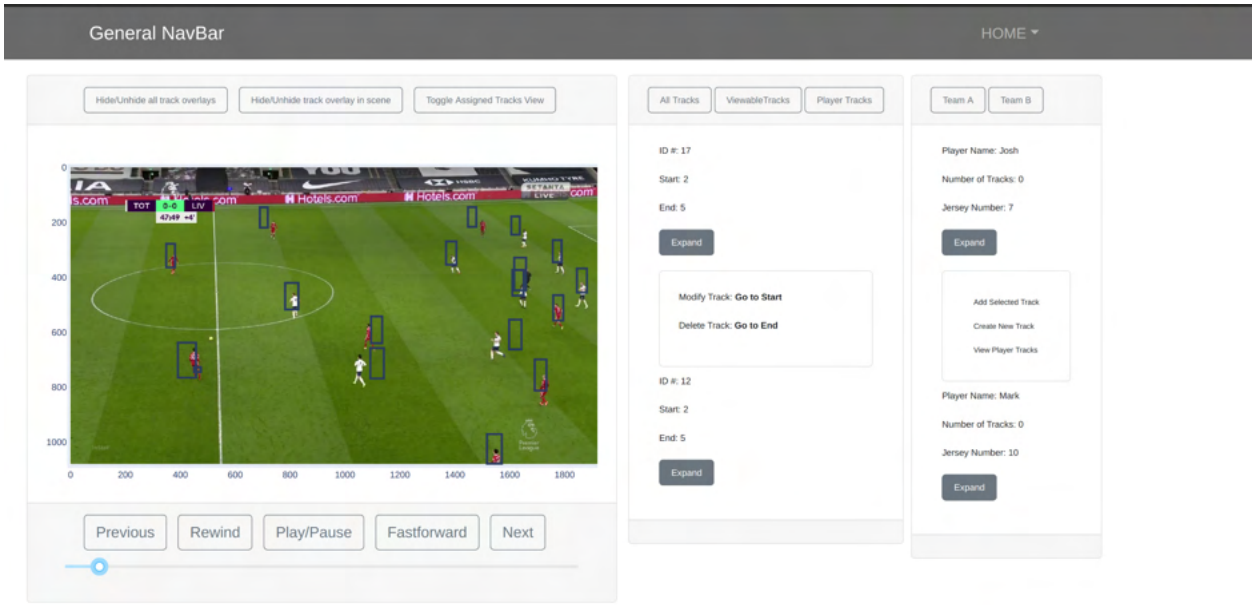


Figure 55: Annotation Tracks Dashboard Implementation

7.3 Web Application Search Engine (FUTURE)

One of the main goals of our projects is to implement a search engine so that the outside users (coaches, managers, players) can upload videos so that they can be analyzed and processed, and an engine to query the already processed video dataset. Since this site will allow for upload we will need a function that checks the type of file the user is trying to upload to be processed. We will only be accepting video files.

The other section of the web application will feature a search engine that retrieves video files and can display videos, which can then be watched by a user.

It will consist of a search bar where the users can enter in a search query to find the video they would like to watch. The user will be able check different properties and refine the search results.

Since we are using S3 to store our video, and Amazon CloudFront now supports streaming of media files stored in Amazon S3 we can use CloudFront to improve access to our static content (HTML, CSS, JavaScript, and so forth) as well as your streaming content.[18]

7.4 AWS

Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. With data center locations in the U.S., Europe, Brazil, Singapore, Japan, and Australia, customers across all industries are taking advantage of the following benefits:

- Low cost
- Agility and instant elasticity
- Open and flexible
- Secure

Amazon Web Services also provides flexible solutions to multiple challenging issues that you may encounter when trying to launch your application. It centers all the necessary services for a successful application. The solutions it offers are:

- Application hosting
- Backup and storage
- Enterprise IT
- Content Delivery
- Databases

Our project encompasses many different services:

- A few machine learning models
- Website platform
    - Front-end
    - back-end
- Database

In order to support all of these services we have decided to use Amazon Web Services. The fact that AWS offers all of the services we need for our application made it an easy choice. In addition, the low price, and the free two month of computation power attracted us as well.

The AWS services we will be using are as follow:

- Elastic Compute Cloud (EC2)
    - Virtual machine
- Elastic Beanstalk
    - Deploying and scaling web applications
- Lambda
    - Serverless automated computation
- Amazon Relational Database Service (RDS)
    - Relational database
- Simple Storage Service (S3)
    - Storage of media files
- Identity and Access Management (IAM)
    - Management of services
- Cognito
    - User access and authentication
- CloudWatch and Simple Notification Service (SNS)
    - Keep track of billing
    - Maintaining the AWS services the application will be using
- SageMaker
    - Implementation of machine learning models
    - Deployment of machine learning models

7.5 Web Application

The web application will start with Amazon's Platform as a Service offering of Elastic Beanstalk. Beanstalk will be able to quickly deploy and manage the application automatically from capacity provisioning, load balancing, auto-scaling, and health monitoring.

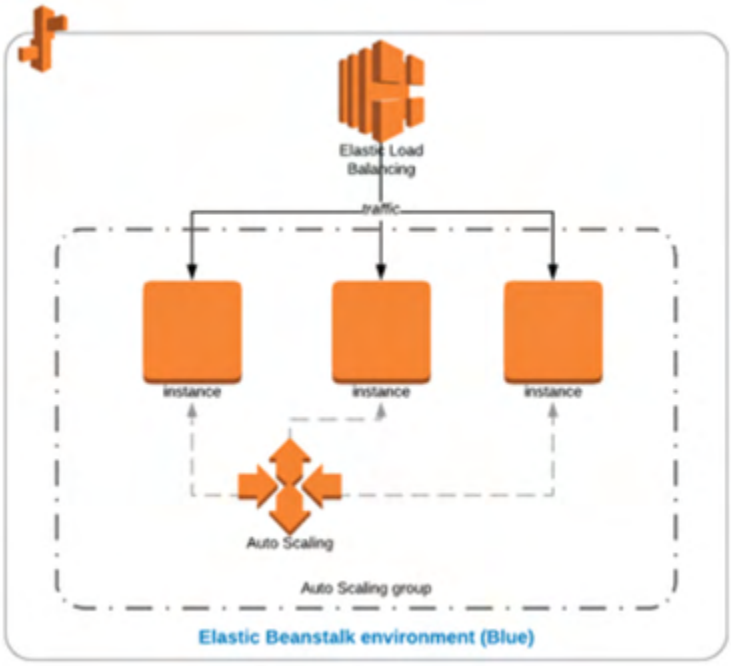


Figure 56: Elastic Beanstalk environment

Here an Elastic Cloud Computing (EC2) instance is automatically created to handle the cloud computing load of the application. Beanstalk will also allow access to the storage options that are specified, from Simple Storage Service (S3) to Relational Database Service (RDS). S3 will be used to store all game footage that users upload while RDS stores user information through the PostgreSQL database.

7.6 Database

For our database we are using a PostgreSQL relational database deployed through Amazon Web Services (AWS).

7.6.1 Overall Design Summary

As stated above, our database employs a relational, SQL model instead of a non-relational, NoSQL model. A relational database was decided upon because our application will likely be storing hundreds of thousands if not millions of player detections for each game that is processed. Relational databases can be more efficient at vertical scale, plus the enforcement of a strict formatting by the table structure for each entry is very convenient.

The core structure of our database is divided into three main parts: Users and Admins, Teams and Players, and The Game and it's inherited tables. Each of these will be discussed at length in the following sections, however a brief summary outline of each section is as follows:

- Users and Admins
    - User
        - This is the basic front-end user of our application. They have the ability to upload games for review and processing and query the database for video clips.
    - Admin
        - This is the user type that is allowed to access any part of the system. These types of users will be able to process games and modify game game data.
- Teams and Players
    - Team
        - Team represents the entries for teams playing in the games. This table would hold the basic information of all teams in the league whose videos could possibly be uploaded and processed. Teams also hold references to their players.
    - Players
        - Players are a subset of teams. Each player has a unique ID and belongs to a certain team.
- The Game and it's Inherited Tables

- Game
  - Each entry in the game table represents one game that has been uploaded to the application for processing. It contains some basic information about who's playing, when it was played, and if the game has been processed or not.
- Detections
  - Each game will have many entries in the detections table for each frame and each player detected in that frame. Each entry includes the frame it was detected on as well as the location for that individual detection.
- Homography
  - There will be one entry per frame of a game in this table. It stores a file with the relevant camera homography information for that frame.
- Field Locations
  - This stores the location of each player at each frame in reference of an actual top-down 2D field view. It's translated from the detections and the homography for each frame and detection.

7.6.2 Users and Admins

The standard user for our system (also referred to as customer) will have basic information and be able to upload video files of games for review. The ID of the games that they upload will be stored in their data as well in order to be able to reference which user uploaded which game. However, it's important to note users do not directly inherit or own games they upload to the system, rather they only contain references to the IDs of the games they uploaded. Any user would be able to access any uploaded game in their queries. In Figure 57, below a diagram represents the table structure of the user object stored in the database.

- User:
  - User ID: An integer representing the user's unique ID.
  - Username: A string representing the user's account name.
  - Password: A string containing the user's hashed password.
  - Email: A string containing the user's email.
  - Name: A string representing the user's name.

- Team ID: An integer representing the user's team affiliation (if any).
- Game IDs: An integer array containing a list of game IDs that the user uploaded.

Admin accounts on the other hand do not upload games. Instead these accounts are the ones that will actually be processing the uploaded games. As such they do not need a team ID or stored list of game IDs. In Figure 57, below a diagram represents the table structure of the admin object stored in the database.

- Admin
  - User ID: An integer representing the user's unique ID.
  - Username: A string representing the user's account name.
  - Password: A string containing the user's hashed password.
  - Email: A string containing the user's email.
  - Name: A string representing the user's name.

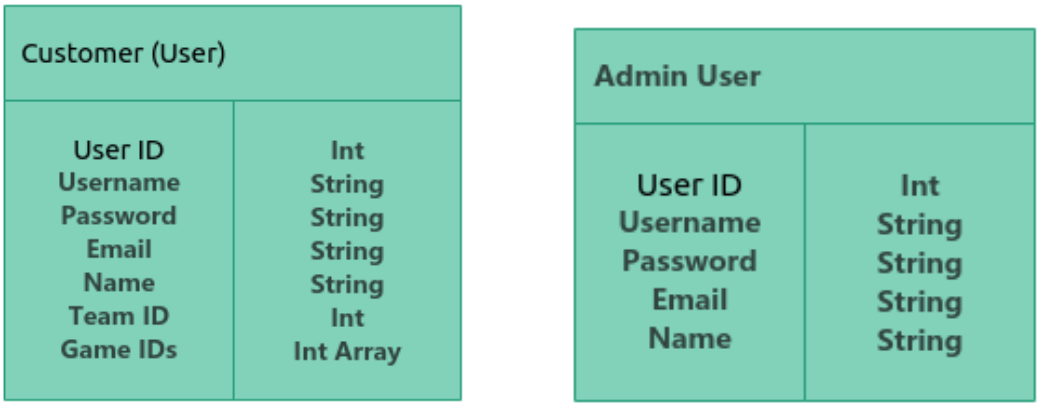


Figure 57: Diagram of User and Admin Structure

7.6.3 Teams and Players

Team represents all of the teams that are playing in the league and whose games could possibly be uploaded and analyzed. If a team doesn't exist in the database when a game featuring that team is added it will then need to be added to the database. Teams also hold information about what players belong to that team.

- Team
    - Team ID: Integer representing the unique ID of the team.
    - Name: String containing the team name.
    - Player ID List: Integer array containing the IDs of all players on the team.

Player represents each individual player that is in the league. Each player belongs to a team and thus has a reference to the team they belong to in the form of a Team ID. Figure 58 below showcases the entity ownership relationship between team and player.

- Player
    - Player ID: Integer representing that player's unique ID.
    - Name: String containing that player's name.
    - Team ID: Integer containing the team that the player belongs to.

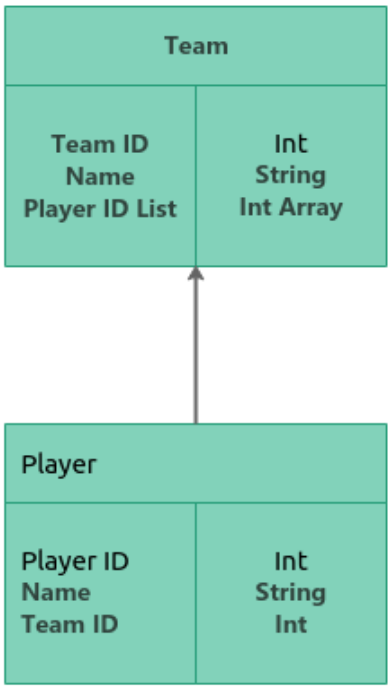


Figure 58: Diagram of Player and Team Structure and Inheritance

7.6.4 The Game and it's Inherited Tables

Each entry in the game table represents a single game that has been uploaded to the application for processing. Each game also has many entries in multiple other tables which are related to the processing of a game. Figure 59 at the end of this section displays the entity ownership relationship between the game table and the three other

tables. Each game holds some meta information about the teams that played and when the game was played as well as a reference to the user that uploaded it. In addition, each game has its own unique ID which is used in the other tables in order to reference which game the entry in that table belongs to.

- Game
    - Game ID: The integer unique game ID.
    - User ID: The integer user ID that uploaded the game.
    - Team 1 ID: The integer ID of the first team playing.
    - Team 2 ID: The integer ID of the second team playing.
    - Day Played: The date of the game played.
    - Video: A large object storing the video file.
    - Processed: A boolean representing if the video was processed yet or not.

The detections table is by far the most robust table of the database, likely holding well over a million entry rows per game. Each entry in this table represents a single detection from a single frame from a single game. Entries have many relations that can be dynamic and change at will, but are designed in a way that writes to the database can always be undone. Each detection is related to a detection track and each track can be assigned to a player. If a track is assigned to a player the player ID value becomes equal to the player's ID. Unassigning is also easy and is done by simply setting the assigned player to -1, which is an ID that no player should have. Through this method none of the original data should be overwritten or destroyed, only new data is added to assign ownership of each entry.

- Detections
    - Game ID: The integer unique game ID.
    - Frame: The integer frame for the detection.
    - x0, y0, x1, y1: Each of these are integers that represent one of four coordinates that make up a detection bounding box.
    - Track ID: The unique per game integer ID of the detection track.
    - Player ID: The unique ID of the player assigned to this track. If there is no player assigned yet the value defaults to -1.

The homography table stores pytorch files for each frame of each game. Whenever a detection is ready to be converted into a field location for a player the homography for that relevant frame is used to accomplish this.

- Homography
    - Game ID: The integer unique game ID.
    - Frame: The integer frame for the homography.

- Homography: A bytea representing a file.

The field locations table stores the location of each player at each frame in reference of an actual top-down 2D field view. It's translated from the detections and the homography for each frame and detection.

- Field Locations
  - Game ID: The integer unique game ID.
  - Frame: The integer frame for the field location.
  - x, y: Each of these are integers that represent one of two coordinates that make up a field location.
  - Player ID: The unique ID of the player corresponding with this location.

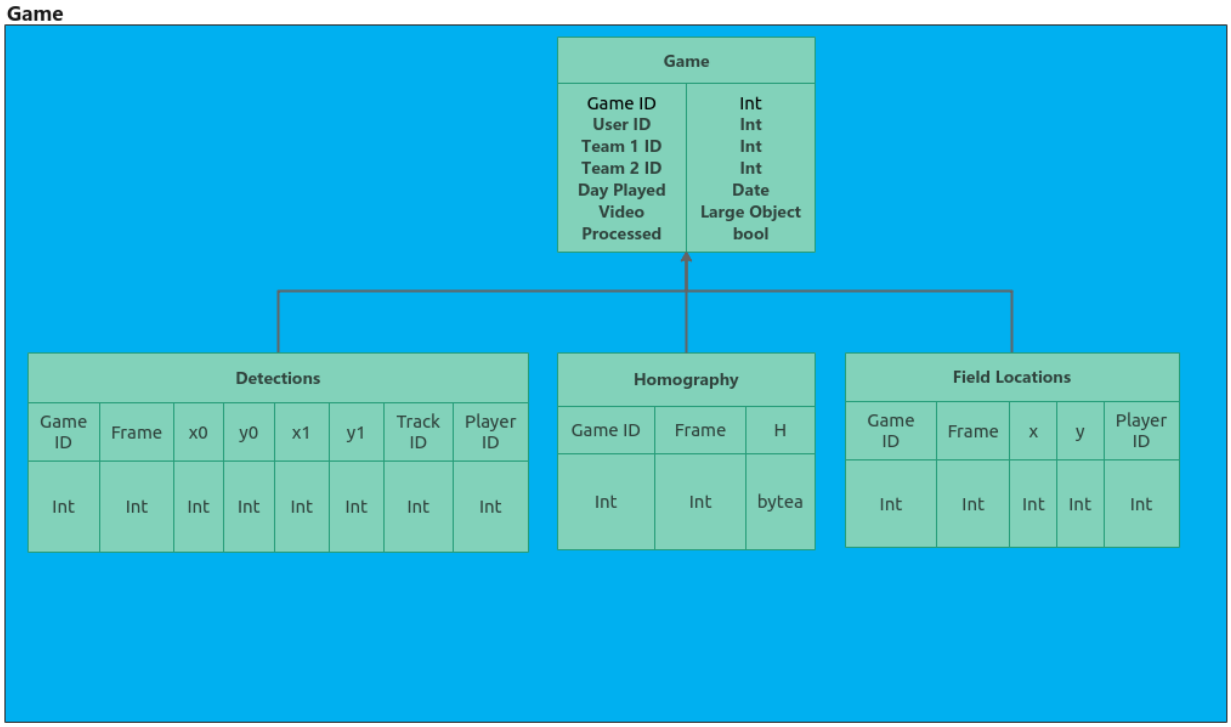


Figure 59: Diagram of the game database structure and all inherited tables.

8 Project Figures and Diagrams

Project Workflow Diagram

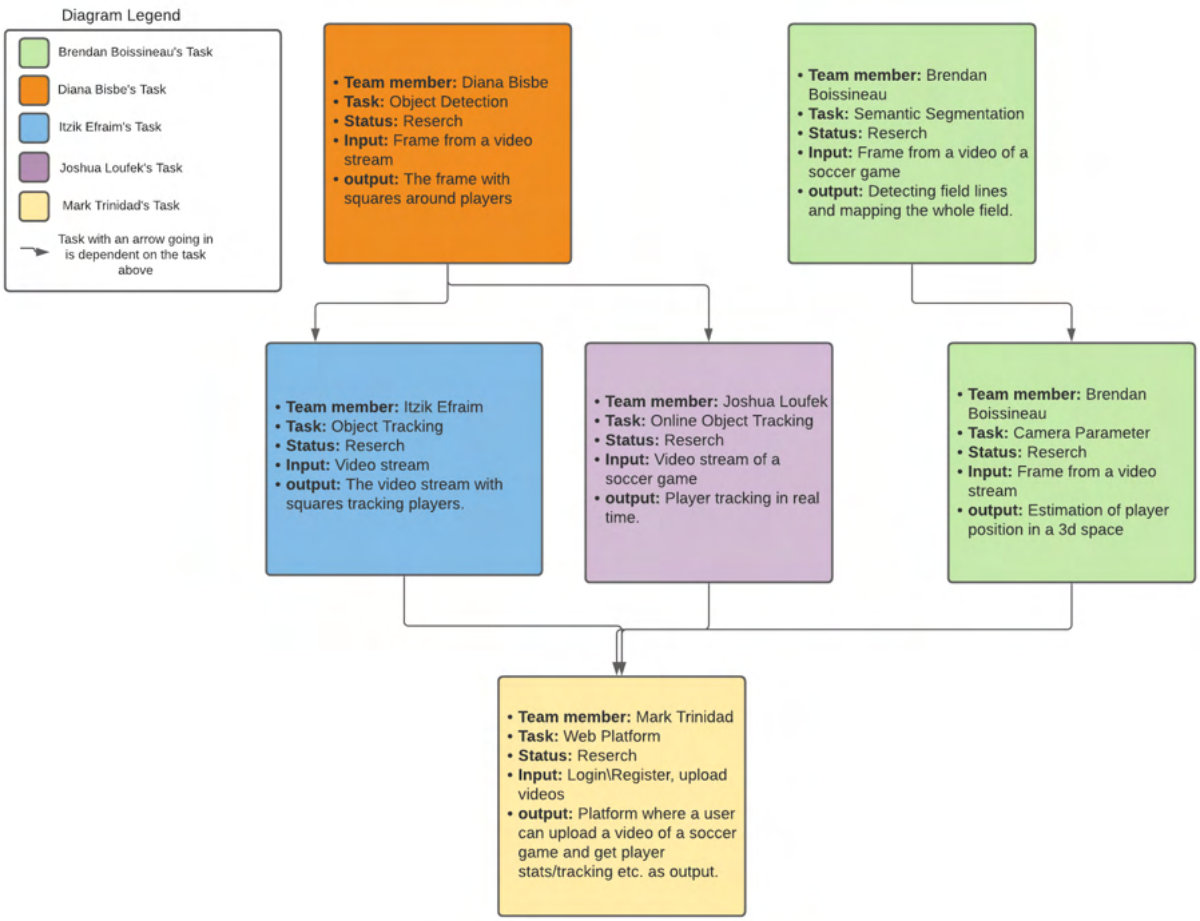


Figure 60: Project Workflow Diagram

System Design Diagram

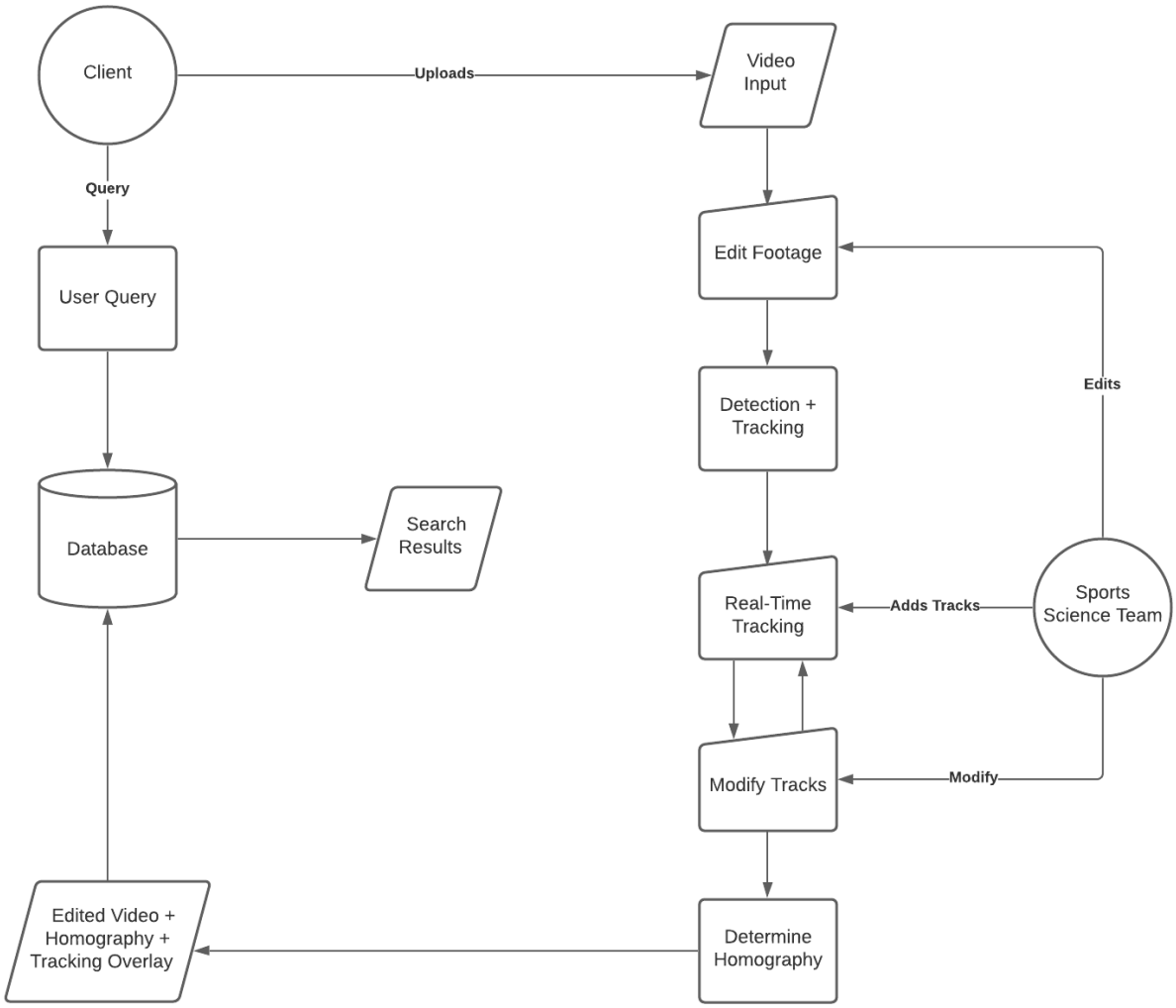


Figure 61: System Design Diagram

9 Budget and Financing

Project budget and financing. Remember to think about potential software licensing costs, charges from cloud-based services (e.g. AWS), mobile app development, code repositories, 2D/3D assets, or website graphic design costs.

A List of Prices and AWS services that we will be using based on US East (N. Virginia) servers:

- Amazon EC2: Virtual machine instance. Pricing depends on resources needed.
    - Example prices:

Name	vCPUs	RAM (GiB)	On-Demand Price/hr
t2.small	1	2	$0.023
t2.medium	2	4	$0.0464
t2.large	2	8	$0.0928

- PostgreSQL through Amazon RDS
    - For each DB instance class, Amazon RDS gives you the ability to specify or provision the I / O capacity that your database requires. You can provision and scale from 1,000 IOPS to 80,000 IOPS and from 100 GiB to 64 TiB of storage with the PostgreSQL database engine. Note that the maximum provisioned IOPS limit will vary based on database workload. With Provisioned IOPS (SSD) storage, you will be charged for the throughput and storage volume that you provision. However, you will not be charged for the I / O operations that you consume.

Storage fee	$ 0.125 per GB per month
Provisioned IOPS rate	$ 0.10 per IOPS per month

- Amazon S3. Database system that we will use to hold video.
    - Standard Storage Pricing:

First 50 TB / Month	$0.023 per GB
Next 450 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB

- AWS Lambda.

Requests	$0.20 per 1M requests
Duration	$0.0000166667 for every GB-second

- AWS Cognito.

Pricing Tier (MAUs)	**Price per MAU**
50,001-100,000 (after the 50,000 free tier)	$0.0055
Next 900,000	$0.0046
Next 9,000,000	$0.00325
Greater than 10,000,000	$0.0025

- AWS SageMaker

Amazon SageMaker capability	Free Tier usage per month for the first 2 months
Amazon SageMaker Studio notebooks , On-demand notebook instances	250 hours of ml.t3.medium instance on Studio notebooks OR 250 hours of ml.t2 medium instance or ml.t3.medium instance on on-demand notebook instances
Amazon SageMaker Data Wrangler	25 hours of ml.m5.4xlarge instance
Amazon SageMaker Feature Store	10M write units, 10M read units, 25 GB storage
Training	50 hours of m4.xlarge or m5.xlarge instances
Inference	125 hours of m4.xlarge or m5.xlarge instances

Website Graphic Design Costs:

- Flaticon. It is a database with a catalogue of vector icons directly to Photoshop. It has a tool that converts icons into web fonts.
    - Cost: $11.99/month
- Freepik. A resources site for vector graphics, stock photos and PSD files.
    - Cost: $14.99/month

10 Milestones

Senior Design 1 Milestones

Date	Goal	Status
February 18	15-page first draft due	Completed
February 21	Finish Tracking Algorithm Research	Completed
February 28	First Algorithm Implementation	Completed
March 7	Create Dashboard Design + Functionality Mockup	Completed
March 14	Finalize Algorithm Implementation	Completed
March 14	Finalize Dashboard Design	Completed
March 15	75-page design document due	Completed
March 28	Setup Video Player w/Track Visualization	Completed
April 4	Setup Mock Dashboard through Dash	Completed
April 4	Finalize Database Design	Completed
April 11	First Iteration of Database Implementation	Completed
April 18	First Iteration of Dashboard Functionality	Completed
April 25	Second Iteration of Dashboard Functionality	In Progress
April 25	Begin AWS Research and Implementation	In Progress
April 28	150-page final design document due	Completed

Senior Design 2 Milestones

Date	Goal	Status
May 9	Final Database Implementation	In Progress
May 9	Start work on User Query View	Upcoming
May 16	Final Functional Iteration of Dashboard	In Progress
May 16	Dashboard Crew Start Working on Linking Web Pages Together and Designing Misc Webpages	Upcoming
May 23	First Functional Iteration of User Query View	Upcoming
May 30	Second Functional Iteration of User Query View	Upcoming
Late May - Early June	SD2 CDR Presentation	Upcoming
June 6	Final Iteration of User Query View	Upcoming
June 13	Finish Linking Web Pages Together and Most Misc Webpages	Upcoming
June 13	Start Migrating Work to AWS	Upcoming
June 20	Start Refining Webpages	Upcoming
June 27	Finish Migrating Work to AWS	Upcoming
July 4	Start Fixing Bugs and Testing	Upcoming
Early July	Heinrich Demo (Mostly done project besides bug fixes)	Upcoming
July 11	Bug Fixes and Refinement	Upcoming
July 18	Bug Fixes and Refinement	Upcoming
Late July - Early August	Final Project Due	Upcoming

11 Project Summary and Conclusions

Looking to the future and imagining our final project, we strive to have a system that is built from a few different parts that communicate with each other. The general idea of our project is a web-portal that can be used by coaches to process their soccer games and search for patterns in other teams' games. In order to do so, we need the following:

- Machine learning models
    - Process the initial video
    - Detect and track players
    - Homography calculation to transform the input video into a 2D soccer map
- Databases
    - Database to store web-portal information
    - Database to store images and videos
- Web-portal
    - Different account types
        - Coaches
        - Video processing team
    - Option to watch the processed video and change wrong detections

Our vision is that eventually this system will serve as "Google" for soccer tactics.

One of the most challenging things about projects that involve multiple models and technologies, is creating a pipeline that makes sense and works efficiently. For that, we have decided to use the services of Amazon Web Services; a cheap, scalable, and secure solution.

The beginning was not easy, as the members of the team were not familiar with the technologies we need to use in this project. Given that, in the last few months, all of the team members have spent multiple hours a week learning the required information. To make it easier, we divided the workload such that every person will only have to learn a couple new technologies. Every sunday we had a brainstorming meeting where each of us presented the research and development they had done during the week. These meetings were useful, as they pushed us to bring results, ask questions, and update one another on the progress being done.

During this time, we have learned the true meaning of trial and error, good communication, and effective team work. Each one of us implemented a number of different models until we found the most relevant one. We have all encountered many difficulties, but luckily we had the support of the team when we needed it.

One of our goals was to make our project scalable. We would want to expand it into different sports, and have multiple users using it. Thanks to the good design, and the abstraction we used, together with the highly scalable Amazon Web Services, it will be relatively easy for us to extend our project and make it usable to many users at the same time.

Throughout our process, we documented every meeting, failure, and success. We kept track of the different technologies we tried to use, and those we will be using in the final product. This document not only represents our final product design, but also the hard work and the process each teammate has experienced throughout the semester.

12 References

[1]
Sharing Data Between Callbacks
https://dash.plotly.com/sharing-data-between-callbacks

[2]
Plotly Python Open Source Graphing Library
https://plotly.com/python/

[3]
Perin, Charles & Vuillemot, Romain & Stolper, C. & Stasko, J. & Wood, J. & Carpendale, Sheelagh. (2018). State of the Art of Sports Data Visualization. Computer Graphics Forum. 37. 10.1111/cgf.13447.

[4]
Chang Ho Ahn. Sports Statistics That Bring Victory. Korea Foundation for the Advancement of Science and Creativity.

[5]
Relational Database Model. (n.d.). Retrieved from http://diranieh.com/Database/RelationalDatabaseModel.htm.

[6]
How to choose the right database model. Marcus Vieira. (2020, February 9). https://www.marcusvieira.tech/how-choose-the-right-databas-model.

[7]
Relational Data Model in DBMS: Concepts, Constraints, Example. Guru99. (n.d.). https://www.guru99.com/relational-data-model-dbms.html.

[8]
Smallcombe, M. (2021, March 9). SQL vs NoSQL: 5 Critical Differences. Xplenty. https://www.xplenty.com/blog/the-sql-vs-nosql-difference/.

[9]
What Is A Non-Relational Database? MongoDB. (n.d.). https://www.mongodb.com/non-relational-database.

[10]
Vitaliy Ilyukha (2020, December 21). Difference Between Relational vs. Non-Relational Database. Jelvix. https://jelvix.com/blog/relational-vs-non-relational-database.

[11]
Research Defence Society. (2007). RDS: understanding animal research in medicine. Amazon. https://aws.amazon.com/rds/.

[12]
Smallcombe, M. (2019, June 14). PostgreSQL vs MySQL: The Critical Differences. Xplenty.https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/.

[13]
Agarwal, Malay. Pythonic Data Cleaning With Pandas and NumPy https://realpython.com/python-data-cleaning-numpy-pandas/

[14]
Jiang, W., Higuera, J., Angles, B., Sun, W., Javan, M., & Yi, K. M. (n.d.). *Optimizing Through Learned Errors for Accurate Sports Field Registration*. https://arxiv.org/pdf/1909.08034.pdf.

[15]
Chen, J., & Little, J. J. (n.d.). *Sports Camera Calibration via Synthetic Data*. https://arxiv.org/pdf/1810.10658.pdf.

[17]
Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (n.d.). Rethinking Atrous Convolution for Semantic Image Segmentation.
https://arxiv.org/pdf/1706.05587.pdf.

[18]
Barr, Jeff. (2009, December 15th) Amazon CloudFront Now Supports Streaming Media Content. AWS News Blog.
https://aws.amazon.com