CS 766 Computer Vision Final Project Milestone

Saikat Raphael Gomes
University of Wisconsin, Madison
saikat@cs.wisc.edu

1. Work done so far:

So far I have started collecting the data required for this project and started processing and extracting feature set. The data collection and processing is explained in details in the following sections.

1.1 Data Collection

I am using Madden 08 PC version to generate the data. Open Broadcaster Software v0.622b – 32bit which is freely available open sourced software, is used for screen capture. A certain play is selected from the existing play book (example: Single back Big TE option, Post Flags route etc.) and the play is executed in the practice mode and then instant replayed from the bird's eye view angle and the above screen capture software is used to records each plays as a video of about 5-8 seconds. This data capture is done on my personal Windows 8 laptop.

I have also created some custom plays and captured and added them to the dataset. So far I have about 100 individual videos.

The eventual goal is to have at least different types of plays with at least 100 samples for each, i.e. a minimum of at least 500 video samples. Currently I am primarily collection single offense plays (i.e. no defense).

Since I expect to collect a large amount of data, I have requested (and got permission [CSL#523968]) for an extra 50 GB of disks space on the AFS drive.

Some of the observations/problems I have faced so far:

- Learning to play Madden: It was quite a challenge to learn to play the game on a PC.
- Using the screen-capture software: there is a lag of about 1-2 seconds between initiating the command to start recording and the actual recording starting. As a result initially I had to discard about 80 samples because the recording started 1-2 seconds after the ball had been snapped, as a result the WRs might have already covered 33% of the visible route. After accommodating for the lag, the video capture is now less of a hassle.
- I have experimented with different field texture and the initially it seems so far the tracking algorithm is indifferent of this feature. But videos with lower texture tend to be of smaller size and hence it makes sense to use low texture to reduce computation time.
- I have also experimented with different teams i.e. different jersey colors. Initially with an older (and naïve) implementation of the algorithm it seemed liked contrasting colors to green gave better results. But with the newer implementation (see below) the jersey color does not matter. But for consistency purpose I am using the Green Bay Packers as offense and the New York Giants as the defense.

1.2 Batch Processing Architecture

The data collected is pushed to a git repository and then exported to the AFS directory mentioned above into a specific data location. Automated scripts are created that wakeup at regular intervals and distribute the new incoming unprocessed data into multiple processing bins for the processing job to process them.

A batch processing script is created that polls the processing bins mentioned above and starts processing the raw data if they are found.

Once a specific data sample is processed it is moved into another specific processed data location.

The above processing is done in parallel manner to reduce the total time take to process the dataset. I am still working making this part automated, but some of the issues with Matlab (below) are causing some roadblocks.

One of the biggest challenges to this part was processing the data in a distributed manner. To overcome this problem, I use the 2-job solution: one job distributes the incoming data into different bins, second job (multiple jobs that are running of different machines) process these bin.

There were and still exist many technical issues. Initially the MP4 codec was missing from the Linux machines [CSL#523680]; this issue was fixed by CSL. Also there seems to be a limited number of Matlab video toolbar license [CSL#524029]; hence I am limited to max of 5 simultaneous processing jobs. Another roadblock so far is we cannot create MP4 videos from Linux [CSL#524026]; this issue cannot be solved and as a result I have to do the processing on a windows machine. I am currently looking into create batch jobs through HTCondor.

1.3 Methodology

Pipeline established so far:

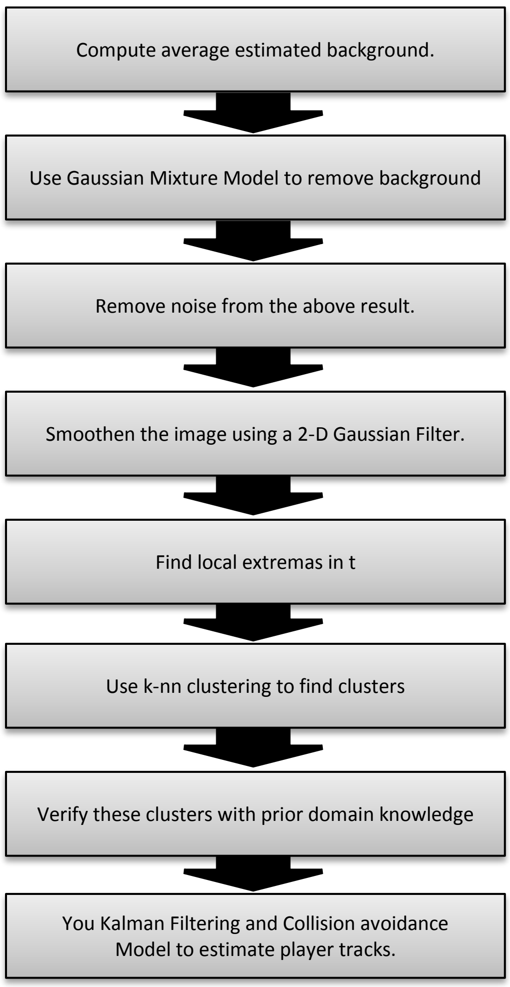

Compute average estimated background.
Use Gaussian Mixture Model to remove background
Remove noise from the above result.
Smoothen the image using a 2-D Gaussian Filter.
Find local extremas in t
Use k-nn clustering to find clusters
Verify these clusters with prior domain knowledge
You Kalman Filtering and Collision avoidance Model to estimate player tracks.

Figure 1: Project pipeline.

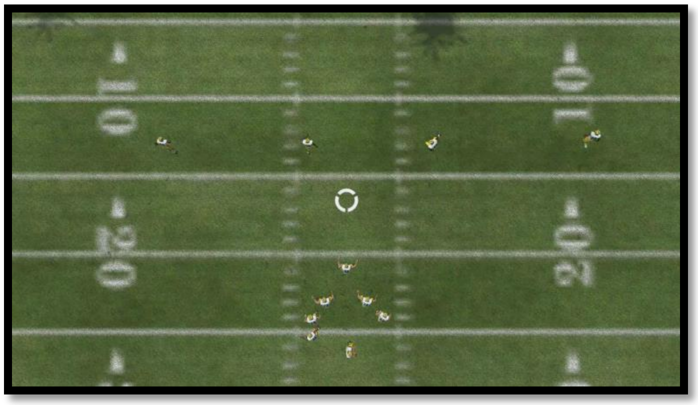


Figure 2a: A screenshot from NFL Madden that will be used as dataset for this project.

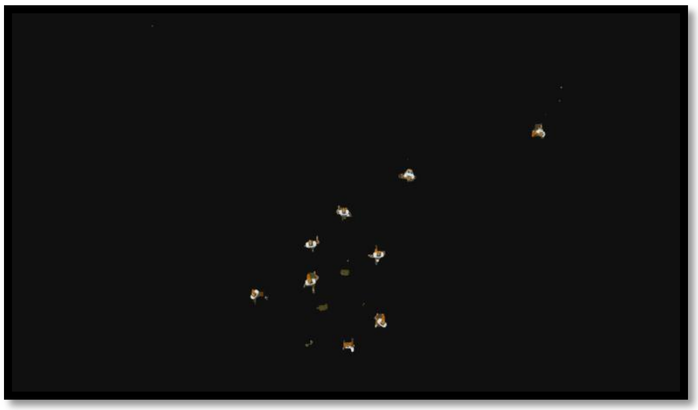


Figure 2b: Background elimination based on Gaussian Mixture Model.

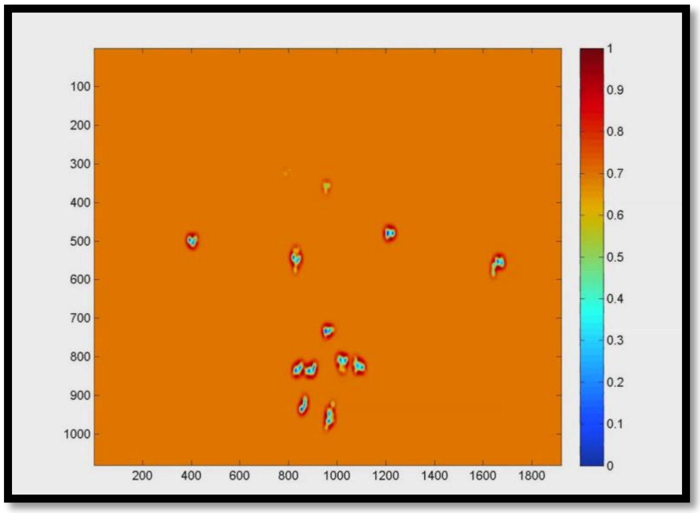


Figure 2c: 2-D Gaussian Filter Smoothening.

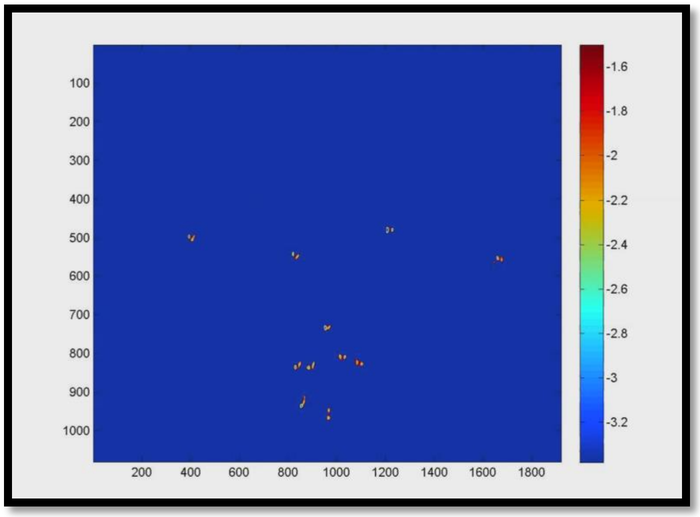


Figure 2d: Thresholding to further reduce noise .

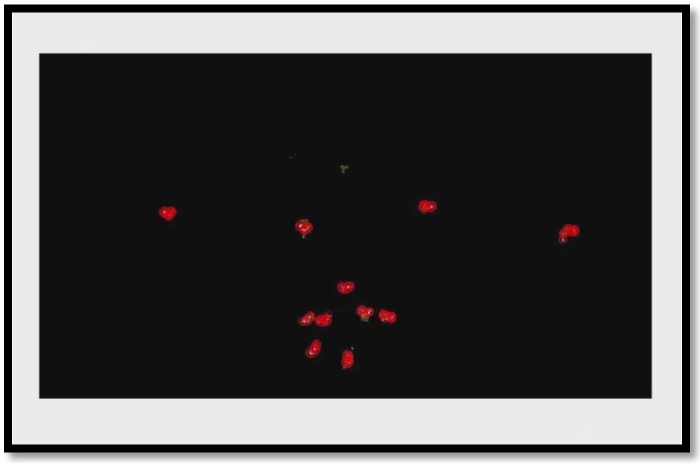


Figure 2e: Computing local extermas.

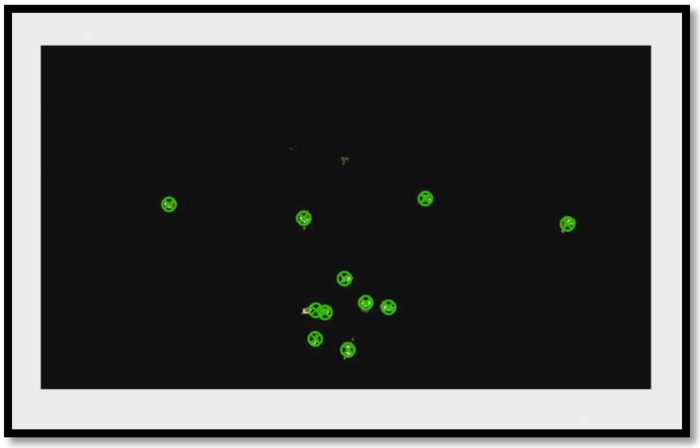
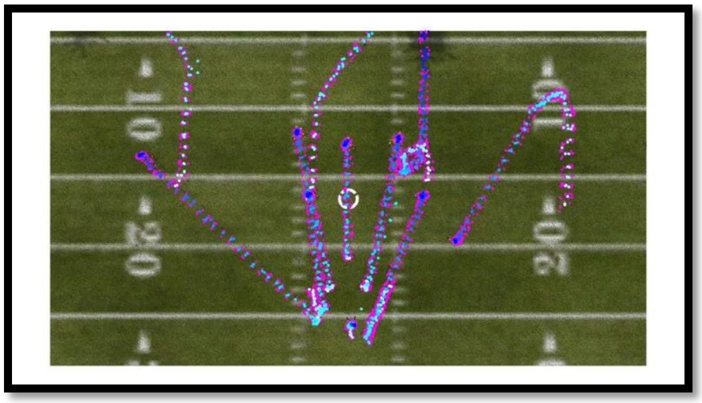


Figure 2f: Clustering to find estimated player location.



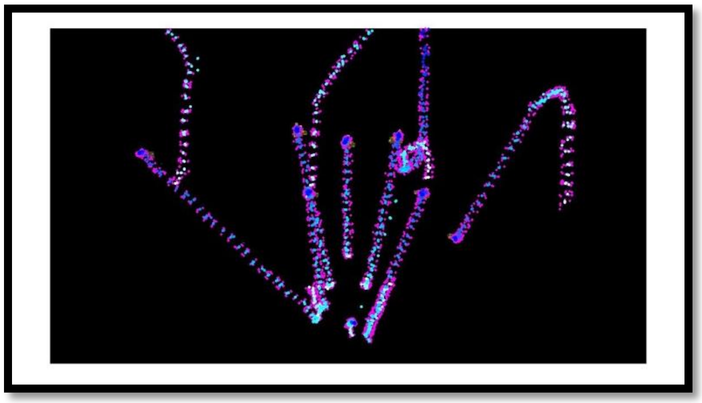


Figure 2g: Tracking of all the paths.

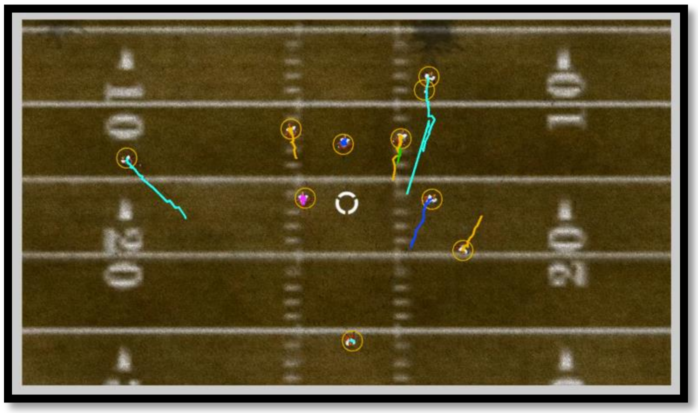


Figure 2h: Using Kalman Filtering to estimate the path of individual players.

Currently I am working on implementing Kalman Filtering to track the individual players in a more robust manner. I am combing some domain specific information to make the method stable.

2. To Do

Once I have implemented the Kalman Filter to track the individual players I have following things to do:

2.1 Use HTCondor for batch processing

I am working on submitting jobs to the windows machines though HTCondor to circumvent some of the Linux issues described above.

2.2 Continue Collecting Data

As mentioned before I am planning to collect at least 500 video samples for this project. This will be a continued till the end of the project.

2.3 Collision Avoidance Model

I intend to explore if the implementation of physics based collision avoidance model makes the tracking more reliable.

2.4 Classification

Once I can reliably extract the tracks and built the feature spaces for each individual plays, I will be classifying the given samples as different plays.

I will be exploring the use of Deformable Parts Models to classify the plays. Another approach I am considering is constructing the feature space from the net regions tracked (figure 2g)

Once the feature space is constructed an SVN will be trained for the final classification phase.

3. Samples:

Git repository:

- https://github.com/saikatgomes/CV-Final

Links to processed data samples:

- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ original.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ noBGVid.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ gradient.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ heatMap.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ localMins.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ clusters.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ centroid1.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ centroid2.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ tracks.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ tracks2.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ tracked_paths.mp4
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ heatMapTotal.jpg
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ tracksTotal.jpg
- http://pages.cs.wisc.edu/~saikat/projects/routeTracking/ tracksTotal2.jpg