Group 6

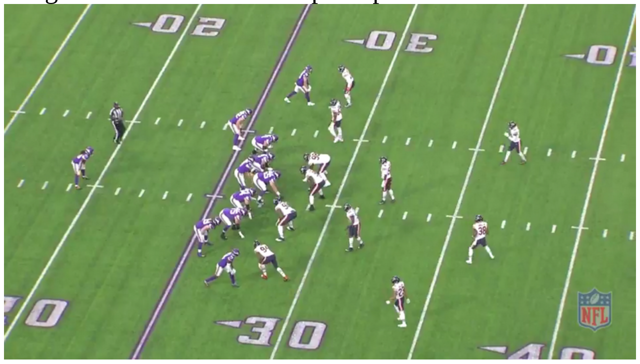
Progress Report: Applying Computer Vision to NFL Game Film for Tracking Player Location on the Field

Peter Wei
University of Michigan
peterwei@umich.edu

Matthew Song
University of Michigan
songmatt@umich.edu

What We Have Completed

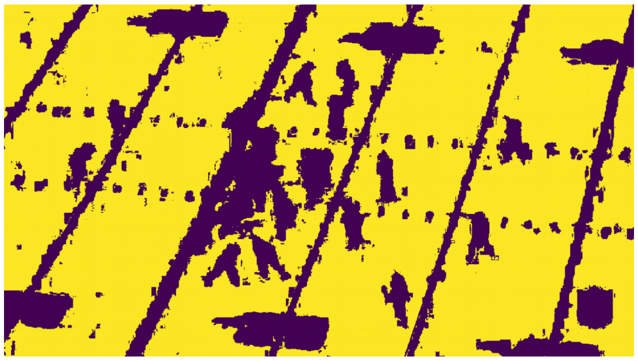
We have completed our implementation of the line detection using the Hough Line Transform. Line detection consisted of two parts: finding the yard lines (continuous, long white lines) and hash marks (small and discontinuous marks perpendicular to the yardlines). The image below shows an example input.



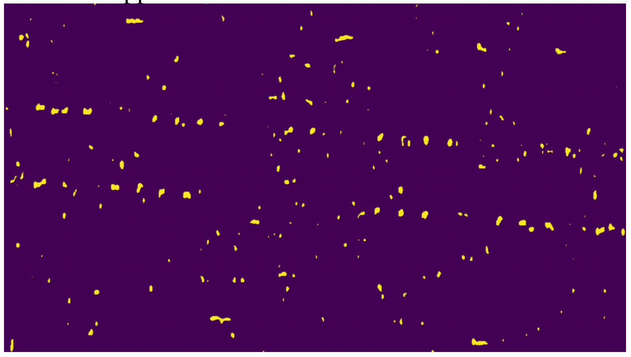
Detecting the yard lines was fairly simple. First, a sobel filter was applied to the image. Then, a Hough line transform was applied to fit lines to the edges. Both these operations were completed using pre-implemented OpenCV functions. Since each yard line has two edges, we also had to merge lines if multiple edges were detected per yard line. This was done by averaging lines with similar angles and offset.

Detecting the hash marks was a more difficult process. First, we tried a similar approach as the one above, but was unsuccessful due to the discontinuity of the hash marks made the line detection algorithm inaccurate and very susceptible to noise from the players. After a lot of experimentation, we decided to use the following process for hash mark detection:

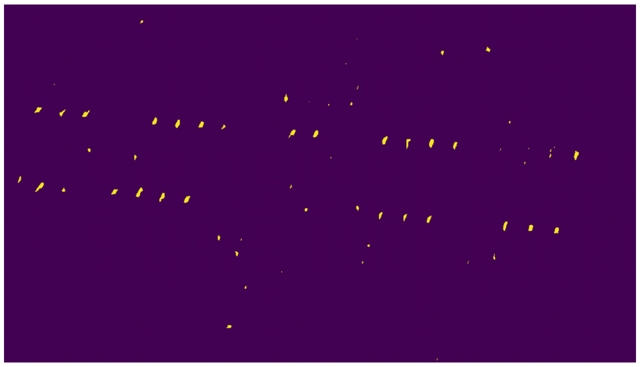
First, try to identify the 'field' parts of the image. This was done by converting the image to HSV and performing a binary filter on the 'H' channel for pixels within the correct green range, and also above a certain saturation value (this removed the white pixels). Below shows the resulting image.



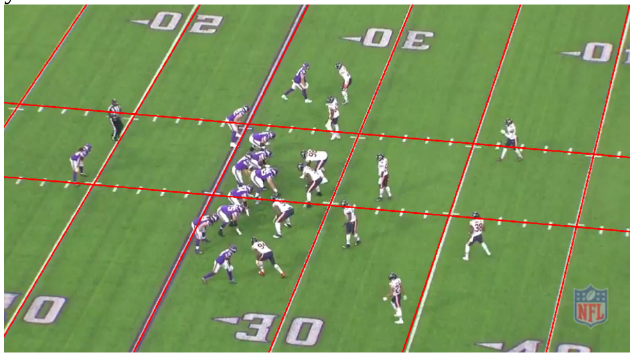
Next, a blob filter was applied. The blob filter was implemented using a Difference of Gaussians kernel with larger sigma in the y direction to account for the elliptical shape of the hash marks. The result was then passed into a binary filter to only keep values above a threshold. Below shows the resulting blob detection after the binary filter was applied.



Then, a mask was applied to the image such that only the slices where hash marks were 'expected' would be kept. These slices could be estimated because the locations of hash marks can be extrapolated based on the locations of the yard lines, since the hash marks appear at regular intervals between the yard lines. Applying this mask reduced the impact of noise on hash mark detection.



Finally, the Hough line transform function was applied to the above image, and duplicate lines were merged the same as with the yard lines. Below shows the result of the yard line and hash mark detection.



Additionally, we have done lots of research on a good algorithm/implementation for player detection on our All-22 images. Some algorithms we have considered are:

- Histogram of Oriented Gradients (HOG)
- Faster RCNN
- YOLO Object Detection (with OpenCV)

Ultimately, we ended up deciding to use Faster RCNN as the base for our player detection algorithm.

We also spend time deciding which game to take our samples from. We decided on the Chicago Bears @ Minnesota Vikings 2018 game primarily due to the following factors:

- The game was played indoors, so lighting would remain fairly consistent throughout the duration of the game and there are no weather effects, allowing for a more consistent data set.
- Both teams are wearing uniforms that are fairly different and have easily discernible features, possible making faster R-CNN detection easier.

What We Have Left to Do

In terms of line detection, we need to still create a function to find the homography between the image and the field. We plan to do this by taking the intersection points of yard lines and hash marks and fitting a homography to the real life distances on the football field.

For player detection, we still need to decide on a pre-implemented faster R-CNN library to use, as well as manually annotate a training set with positive and negative samples. There are a lot faster R-CNN libraries available, and we still have to identify which one we want to use for our project. We want our training set to contain abou 1000 positive examples of players, which we will manually annotate from game film still images, as well as a large number of negative examples. This will translate to having to grab about 50 still images from a single game of All-22 film, as each image will have 22 players, and 50 images will result in around 1000 player snippets. We believe getting the algorithm to train correctly and accurately will be the most difficult challenge for the rest of the project, but we are confident that upon its completion, we will be able to accurately detect all players from an input All-22 game image we give it.

If time permits and we are able to get our Faster RCNN to work accurately, we would like to attempt to translate our player detection algorithm from still images to dynamic video. We believe this should be a fairly simple task, as the main goal is to be able to track players and field lines as the camera perspective moves, and we can simply apply our still image player detection algorithm to each frame of the video. However, this task could be more difficult if we wanted to implement the feature of tracking individual players, since it would be difficult to match players from one frame to another, especially if the faster R-CNN misses detecting a player for a number of frames.