AN ABSTRACT OF THE THESIS OF

Amit Bawaskar for the degree of Master of Science in Computer Science presented on May 30, 2014.

Title: Interactive Player Tracking in videos of American Football

Abstract approved: _____

Sinisa Todorovic

This thesis presents an interactive software tool for tracking a moving object in a video. In particular, we focus on the problem of tracking a player in American football videos. Object tracking is one of the fundamental problems in computer vision. It is one of the most important components in numerous applications of computer vision. For our domain of American football videos, there are no existing trackers which are sufficiently robust to accurately track a player, due to many challenges (e.g. occlusion, camera motion). Our interactive software tool for tracking is aimed at improving the performance of a tracker through human corrections of tracking errors. The literature presents a number of similar interactive tools for tracking. However they are not suitable for our focus domain, because they lack in one aspect or another. Some are not real time, some require sanitized settings, some make restrictive assumptions, while some cannot handle long videos. We use the state-of-the-art method called Online Multiple Instance Learning (OMIL) tracker as our base tracker. While tracking a player, our interactive tool allows

the user to tell the tracker the exact location and size of the player. Given this user feedback, the system updates the player's trajectory and places a bounding box surrounding the player in each frame. To ensure smoothness in trajectory, we use the spline interpolation to interpolate the player's trajectory between the user-specified position and the last position estimated as correct. In addition, we estimate the player's acceleration and use it for placing the bounding box along the interpolated trajectory. All this is done at nearly real time. Our large scale experiments on 355 American football videos demonstrate the effectiveness of our tracking tool. We annotated 355 videos with ground truth trajectories for a set of players of interest. We categorized tracking tasks according to player types so as to provide the user with an estimate of the number of interactions required for getting the expected degree of accuracy. A comparison of tracking results with and without our interactive tool demonstrates the increase in accuracy from 62% to 88% on average in the former case on 355 videos. We also demonstrate that the tool is not tracker dependent by testing it with another tracker called Particle Filtering.

Interactive Player Tracking in videos of American Football

by

Amit Bawaskar

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented May 30, 2014
Commencement June 2014

Master of Science thesis of Amit Bawaskar presented on May 30, 2014.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Amit Bawaskar, Author

ACKNOWLEDGEMENTS

I would like to express my special appreciation and thanks to my advisor Prof. Dr. Sinisa Todorovic, you have been a tremendous mentor for me. I would like to thank you for encouraging my research. I greatly appreciate the patience and support you have showed me throughout my time at Oregon State University. Your guidance was instrumental to me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master's study.

Besides my advisor, I would like to thank Prof. Dr. Alan Fern for the continuous support of my research, for your patience, motivation, enthusiasm, and immense knowledge. I appreciate your support, guidance and advise during the last two years.

I would like to take this opportunity to thank my fellow graduate students with whom I have shared many stimulating discussions. It was an immense pleasure to be working and at the same time having fun with you all. You have made my stay here enjoyable.

I would also like to thank my parents. They were always supporting me and encouraging me with their best wishes.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

1.1 Problem Statement

Object tracking is a fundamental problem in computer vision. This problem arises in a wide range of applications including video surveillance and analysis of sports video. In this work, we focus on the domain of football videos where object tracking is used for highlighting players during a play. Player-highlighting is important for estimating a wide range of characteristics of all players. For example, this highlighting is frequently used by coaches and recruiters. From player tracking, they extract statistical data and general information about players' behavior and capabilities. Players often need to track themselves for showcasing their abilities.

In this work, we focus on the football domain. American football is highly structured domain and it presents a rich set of challenges. For this reason, research in this field is of general interest in the computer vision community. In addition, the football domain is of significant practical and commercial interest. Professional football teams and college teams typically employ crews of video scouts whose job is to organize, annotate, and analyze huge collections of football video. Information from these videos is then used for various coaching purposes. There is an opportunity for computer vision to help annotate video analysis in the football domain.

1.2 Challenges

Videos of American football present a number of challenges for existing approaches to object tracking. These challenges arise from camera motion, occlusion, varying player movements, interaction between multiple players, visual similarities between different players, etc, as explained in greater detail below.

Camera Motion: In order to capture the game, the camera pans left and right, or zooms in and out. The cameraman will try to keep the ball and the main players such as quarter back, etc. in view at all times. To do so, there will be significant camera motion which is not typical of other types of video. This camera motion introduces significant motion blur and modifies object trajectories under perspective projection. The state of the art in tracking usually takes into account little to no camera motion. Existing trackers make the assumption that the object being tracked is usually in the center of the frame. This assumption does not hold in football videos. Also, the trackers typically make the assumption about the direction of motion which we cannot make as players can move in various direction on the field depending on their roles eg, offense-defense. Existing approaches typically assume smooth motion trajectories. This assumption does not hold because, often times the camera moves in the opposite direction of the players' motion introducing non smooth motion trajectories.

Occlusion: The sheer number of players and referees on the field at any given time as well as the nature of the football game, where the players clump in large groups make it very likely that a player will be occluded by another player during the course of the play. Some occlusions will be transient but there will be long term occlusions.

Existing approaches typically make the assumption that occlusion is temporary and short term. This assumption does not hold in the case of football videos. Since the tracker is typically unaware of the duration of the occlusion, it finds it difficult to determine if the object in consideration is occluded or has changed appearance.

Varying player movements: In American football, there are various types of players such as quarterbacks, running backs, wide receiver, tackle, guard, line back, corner back, safety, etc. Every player-type has a specific role. The way they move is governed by the role assigned to each player. There are different strategies used by different teams and the movements of the players will be different taking into account those strategies. As a result of this variety, it is very difficult to predict how a player will move in any given situation. Existing approaches typically make the assumption about the behavior of a moving object. For example, they assume constant acceleration, movement in the same direction (e.g. car on the street). How ever this assumption does not hold in American football videos.

Interaction of multiple players at a time: During the play, especially at the start, there is a lot of interaction among the players. Usually, the guards and tackle try to out muscle each other while the quarter back tries to find a player to who he can pass. If there is a running back, he will most probably try to get past the line of tacklers and guards to get as far up the field as he can. This creates interaction among multiple players as players fall, tackle, get tackled, etc. As a result of this, a player that is being tracked may significantly change in appearance and shape. Existing tracking systems typically make the assumption of smooth appearance and shape changes and thus use smooth updates on the object model for tracking. However this assumption does not hold in

our case, and we need to relax the assumption. We require a dynamic object tracking system which will constantly update the model so that any change in the appearance is well noted.

Visual similarity among the players of the same team: Players of the same team have the same uniform. This can confuse the tracker to believe that a nearby player from the same team is the tracked player. Existing approaches typically assume that the tracked object is salient and different from the background.

Due to the aforementioned challenges, developing tracking systems for American football has not had success. A reason for this may be because most of the existing state of the art tracking approaches consider sanitized settings for their specific problems [44, 46, 36, 26, 6, 8, 50, 13, 21, 49, 27]. Such settings include but are not limited to background distinction for the target, no occlusion, etc. Therefore, existing methods cannot robustly overcome these challenges. Therein lies our motivation to create an interactive tool which would be able to provide accurate tracks for the highlighted players with minimal user assistance.

1.3 Overview of the Thesis

Although there are a number of off-the-shelf tracking algorithms, they cannot be readily used for tracking players in American football videos. Therefore, we have developed a user interface for facilitating correction of errors in tracking of players in American football videos. Thus we aim to develop an interactive tool for tracking which improves the performance of the basic tracker via introducing human correction of tracking errors.

As the basic tracker we used, the Online Multiple Instance Leaning (OMIL) tracking algorithm [6]. In our tool, the tracker is initially run on raw videos. The tool helps the user make corrections by restarting the tracker. The main reasons for using this tracker include the following:

1. The algorithm meets out stringent requirement for real time tracking. It is efficient and gives results in nearly real time.

2. Its an online algorithm and is capable of learning the changes in appearance of the tracked player.

3. Babenko et al in [6] have compared the algorithm with several other algorithms in their work. Their experimental results demonstrate that this algorithm is more robust than competing approaches.

4. The algorithm provides a stable path of the tracked object. This is important for us as we have tried to improve the path outputted by the tracker by using the interpolation strategy. This would be difficult by using other trackers, e.g. Particle Filter [25, 35, 48] in which often the path of the tracked object is not smooth.

The major components of our system include: adapting a state of the art tracker and our interactive tool to enhance its performance.

Adapt state of the art tracker: We have adapted one of the state of the art trackers, Online Multiple instance learning tracker, to suit our purpose of tracking players. we made certain software implementations of the algorithm We have modified the algorithm to work well under normal circumstances and track players as long as they are not occluded or out of view.

Interactive tool: The tool enables error correction. Whenever there is an error in the tracking system, the user can pause the tracking and reselect the player from a new position. We have created an interactive system where in the user can decide that the tracker has lost track of the player of interest. At this point the user will pause the video and reselect the player. This feedback from the user will be incorporated into the tracking algorithm so that the user defined feedback is always in the path of the track. The incorporation of feedback into the track will ensure that the track of the player will remain accurate.

Since such feedback will have abrupt jumps in the tracking of the player, we have developed an interpolation technique in the system. This interpolation will correct the error in the track of the player so that a smooth transition of the track is seen from one frame to another. The interpolation technique will dynamically determine the frame where the tracker lost track of the player using the confidence of the tracks in the preceding frames. Then, the system will interpolate between the newly selected user feedback frame and the dynamically determined frame so that such jumps are not observed in the video and a clean and smooth track of the player is output by the system.

We do this interpolation in three parts.

1. Interpolation of trajectory: Here we interpolate how the player must have moved. We take into account the new position from the user feedback and the trajectory of the player where the tracker estimation is confident. This interval is called as the 'Interpolation Window' . We use spline interpolation to determine the trajectory that the player must have followed during this Interpolation Window.

2. Interpolation of acceleration After interpolating the trajectory of the player we interpolate the acceleration of the player in the Interpolation Window. We estimate the acceleration the player must have had taking into account the acceleration before and after this window. Using this information we interpolate the acceleration of the player. We estimate the position for the player for each frame during this Interpolation Window on the interpolated trajectory.

3. Interpolation of the size of the bounding box On determining the position of the player, we interpolate the size of the player in the Interpolation Window. We determine the size of the player in each frame in this window by considering the last confident bounding box and new user defined bounding box and assume a linear change in the size of the bounding box.

We have manually annotated 355 American football videos with ground truth trajectories of a set of players of interest. We use these annotations to determine the boost in performance of the tracker.

1.4 System Block Diagram

Fig.1.1 shows the key components of our tracking. As shown in Fig. 1.1, the user will first start the system by choosing a video. Next the user will initiate the tracking process by choosing a player, by drawing a bounding box in any frame the user wishes. The tracker will then track the player. At any time during the video, if the user feels that the tracker has lost track of the player, the user can reselect a bounding box around the player. The tracker will use this feedback to adjust the players track. The user can see

the interpolation at any time by rewinding the video. At the end of the tracking, the system will output the video having a bounding box around the player at all times.

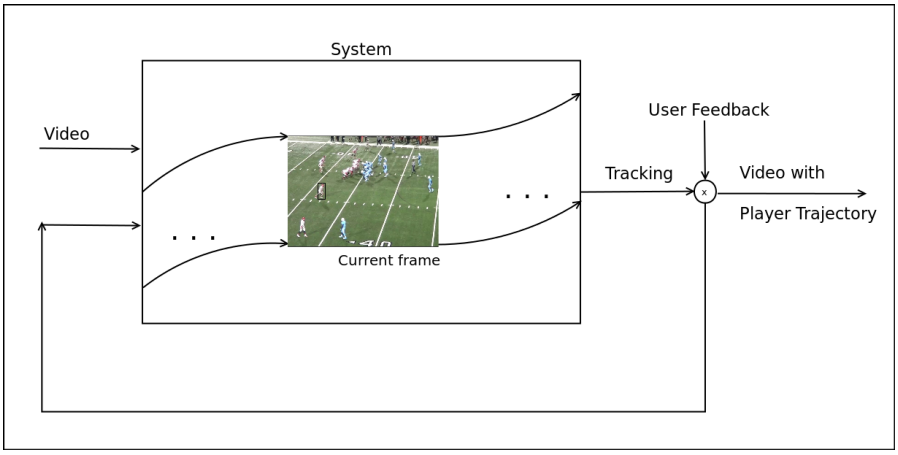

Figure 1.1: Block Diagram of the Player Tracker

The features mentioned above enable it to overcome the challenges faced by the tracker in order to get a good track of any player in a football play. Since the user will be able to give feedback during run time, there is no latency in the processing of the video. The user will be able to get the desired video clip of the tracked player in almost real time. We will actually slow down the video being displayed so that the user will have ample time to provide feedback.

This is one of the first approaches to track football players in real time using user feedback.

Chapter 2: Literature Review

A wide range of computer vision applications require object tracking. Given a target object in the first frame, in particular its position and size, the goal of tracking is to estimate the position and size of the object in the subsequent frames. Object tracking has been studied for several years and much progress has been made in recent years. Yet it remains a very challenging problem. As mentioned in chapter 1.2, there are many challenges involved in object tracking and rarely an approach can claim to successfully handle all of them.

Most approaches for object tracking contain three basic modules: appearance model, motion model and search strategy. We have organized our literature review into groups of tracking approaches that differ in modeling of appearance, trajectories and whether human is encoded in the loop.

2.1 Trackers organized by appearance modeling

In this section we review approaches that use particle filtering, tracking by detection and static or online strategies for modeling appearance.

2.1.1 Particle filtering

Particle filtering is a widely used framework for visual object tracking that is highly extensible and offers the flexibility to handle non-linearity and non-normality in the object models. In recent years, many new particle filter-based approaches have been proposed to solve difficult object tracking problems [25, 35, 48]. However, most of this work has relied on some combination of manual tuning and simple generative learning to set the parameters of the proposed models. These approaches, though, are often ineffective and/or extremely labor intensive.

Since particle filter-based visual object tracking was first proposed, much progress has been made on tracking single objects using particle filters. Particle filter methods can be viewed as driving the learning process by iteratively using the current filter to perform inference on a set of training sequences and then updating parameters based on some measure of the disparity between the filter's output and the desired output. For example, Limketkai et al.s approach [28] computes a maximum a posteriori (MAP) state sequence for each training example using the current filter and then attempts to adjust filter parameters so that the ground truth sequences become more probable than the current MAP sequences. However because filters sometimes fail badly, often early in training, the filter parameter updates these approaches make can be dominated by the portion of the sequence that occurs after the failure rather than focused on correcting the original point of failure. Our experience suggests that such parameter updates can be counterproductive.

To address this issue, a perceptron algorithm has been extended to focus training

directly on points of filter failure. For example, recent work has proposed performing updates based only on the part of the predicted sequence up to and including the first failure [14]. Later work has extended this idea by updating at successive points of failure [47, 16].

However, even with these updates in the general ideas in Particle Filtering, there remains a basic necessity to perform manual tuning and learning to set the parameters of the proposed models for the filter. This leads us to making a complex tracking system which tries to overcome the challenges faced by trackers by creating a complex model of the object. It tries to tackle occlusion and camera motion by learning or tuning the parameters of the filter and trying to figure out when the errors are caused, which is a hard problem.

Though it works well in cases where a smaller number of challenges are present, our experimental analysis shows that the tracker output is not stable where there are multiple challenging cases mentioned in chapter 1.2 at once. This makes it hard to visualize a decent smooth trajectory for players, as in our case of American football. Also, as there are similar looking players on the field at a time in American football, the filter is more prone to failure. Therefore, we cannot use a Particle filter based approach for our needs of tracking American football players.

2.1.2 Tracking by detection

In tracking by detection there is a large set of candidate detections. In the set of candidates we find the target object by linking the most likely appearances in each frame

using network flow [11, 3, 10, 19, 22]. The concept in approaches of this method is to detect the object in every frame. This will give the notion that the object is tracked over the length of the video.

2.1.3 Static Appearance Model

Certain tracking methods of this type [23, 35, 20, 9, 33, 1] employ a strategy in which they train the appearance model of the tracker using only the first frame of the video. To track the object they try to detect that object in the all the frames of the video. This method handles camera motion very well as there is no significance spatial information of the object. Wherever the object is within the frame, it will get detected. But these methods suffer from a drawback. They are unable to cope with any appearance change in the target object. If the appearance of the object changes, then the tracker will look for something which is not present in the frame and will either select something else that looks like the original object or will fail completely. This renders these methods not useful for our purpose as the appearance of the players changes in every frame. They sit, run, change direction at will and a static appearance model cannot be used to track them.

2.1.4 Adaptive Appearance Model

To overcome the change in appearance of the target object, tracking methods which employ adaptive appearance models [31, 24, 37] have been developed. Training such

adaptive methods becomes very difficult as it creates more problems than solutions. For example, the tracker must now know when to update its appearance model and how to deal with partial or full occlusion. For these types of algorithms to work and give accurate results, their parameters must be correctly tuned to the specific problem at hand. Such parameter tuning may give out accurate results in some cases but it may fail miserably in other cases. For example, if we choose to update the model every 5 frames, the tracker will give correct results for tracking any player if the every occlusion in the video lasts for less than 5 frames. It would fail if the occlusion lasted for more frames. On the other hand, if we increase the number of frames when the appearance model is updated to a higher number the object may change multiple times before its appearance model is updated and go wrong. So, deciding such parameters is hard. Also, there may not be a parameter setting which would work for all the videos.

An important choice in the design of appearance models is whether to model only the object [37, 7], or both the object and the background [4, 5, 45, 15, 30, 18, 29]. Many of the latter approaches have shown that training a model to separate the object from the background via a discriminative classifier can often achieve superior results. These methods are closely related to object detection an area that has seen great progress in the last decade. In fact, some of these methods are referred to as 'tracking-by-detection' or 'tracking by repeated recognition' [32]. In particular, the recent advances in face detection [41] have inspired some successful real time tracking algorithms [18, 30].

Object detection faces issues similar to those described above, in that it is difficult for a human labeler to be consistent with respect to how the positive examples are cropped. In fact, Viola et al. [41] argue that object detection has inherent ambiguities that cause

difficulty for traditional supervised learning methods. For this reason they suggest the use of a Multiple Instance Learning (MIL) approach for object detection. Viola et al. [41] present convincing results showing that a face detector trained with weaker labeling (just the center of the face) and a MIL algorithm outperforms a state of the art supervised algorithm trained with explicit bounding boxes.

Since it was most appealing, we tried to use this method to track players in American football videos, but we found out that it suffers in the challenges we mentioned in chapter 1.2. Though it performs well when there is occlusion, it cannot handle the presence of similar objects in the vicinity. It often learns the wrong object quickly and decides to track that instead of the object in question. Since this scenario is often encountered in American football, we cannot use this method alone to track players in American football videos. Therefore, we have tried to extend this approach so that we could generate seamless and smooth tracks for players in American football videos.

2.2 Modeling trajectories

Formulation of tracking as a global path optimization problem has been recent development in this field [2, 12, 39]. Rotoscoping [2] is a contour based tracking designed for graphics applications. The contour tracking is cast as a space time optimization problem that solves for time-varying curve shapes based on input video and user inputs. It is difficult to be applied to other tracking tasks. In [39], 3D curved object trajectory segments are extracted from the video using spectral clustering. Then, the occlusion and final object path are inferred based on these trajectory segments. The trajectory segment

extraction and analysis are too slow (e.g., 1 fps) for interactivity. The interactive feature tracking system in [12] tracks a 20×20 image patch. A k-d tree is used to quickly detect the feature and produce multiple hypotheses in every frame. The detection can run at 100 to 200 fps for a 320×240video. The patch based representation has limited effectiveness on generic objects that can change pose, viewpoint, scale and aspect ratio. The k-d tree will be very expensive to support the patches with varying size and shape. Before the user interaction, the k-d tree cannot be precomputed since the size and shape of the object is unknown. As earlier developed such methods are either too slow or too expensive, we cannot use them effectively to track players in American football. We have however, used this idea of user feedback and applied it to real time tracking systems to get highly accurate and real time tracking of the players.

2.3 User Assisted Tracking Mechanisms

The simplest type of user assisted tracking systems do not actually use a tracking algorithm. These systems ask the user to annotate the object in different the parts of the video. A variety of systems [40, 43, 38] have been developed to annotate video data for research purposes use this approach. Then, they use an interpolation technique to simply ”join the dots” and provide a track for the object. Such systems cannot handle many of the challenging aspects of tracking, especially camera motion as they perform simple interpolation and do not actually track the object. The also cannot handle the varied and erratic motions of the players as they cannot be determined only through annotation. Also, since the user annotates different frames of the video, the user is doing most part

of the work in such user assisted tracking methods. These methods perform better when the user provides more annotated frames, i.e. it requires manual labor to get accurate results. Therefore, these methods too cannot be used for our purpose.

In case of the algorithms mentioned in the chapters 2.1.1 and 2.1.2 and their varieties, accurate tracking is hard to achieve. Such fully automated methods have no way to retrieve a lost object. When application requires high tracking accuracy, an interactive system that involves user inputs is often necessary. A small amount of user assistance can help a tracker to recover from failure. For example, the user can specify the object state (e.g., location and size) in a few frames which can be used effectively to generate a good track.

Chapter 3: Online Multiple Instance Learning Tracker

We have implemented the state-of-the-art Online Multiple Instance Learning (OMIL) object tracking algorithm which can robustly track image patches across a video sequence. It is an online approach that learns the appearance of the object changes from frame to frame. The input to the tracker is a video and an initial bounding box of the target object, defining the first location and dimensions of the image patch to be tracked. On each subsequent frame, the tracker places the bounding box at the location of the target object.

3.1 System Overview

Algorithm 1 OMIL Algorithm

Input: Video frame

1: Crop set of image patches on the object $X^d = \{x : ||l(x) - l_{t-1}|| < d\}$ and compute feature vectors for these patches.
2: Use the appearance model to denote the presence of the object, i.e. estimate $p(y = 1|x)$ for each $x \in X^d$.
3: Update tracker location to the image patch having maximum response from the classifier $l_t = l\big(\arg\max_{x \in X^d} p(y|x)\big)$
4: Crop two sets of image patches on the new object location $X^{pos} = \{x : ||l(x) - l_t|| < r\}$ and $X^{neg} = \{x : r < ||l(x) - l_t|| < \beta\}$.
5: Update appearance model with one positive bag X^{pos} and negative bags X^{neg}.

The key steps of OMIL tracker are given in algorithm 1. The tracker considers a player to be identified by a tight rectangular bounding box around him. The player

representation for tracking consists of a set of Haar-like features [41, 17]. The tracker consider 4 rectangles around each image patch represented by the player, having real valued weights. The feature value is then a weighted sum of the pixels in all of the rectangles. We use a discriminative classifier $p(y|x)$ as the appearance model, where x represents the image patch and y is a binary variable that denotes the presence of the object in that image patch. At every time step t, our tracker maintains the object location l_t . To get the location of the tracker in the next frame, the tracker crops a set of image patches $X^d = \{x : ||l(x) - l_{t-1}|| < d\}$ that are within some search radius d of the current tracker location, and compute $p(y|x)$ for all $x \in X^d$. We then update the tracker location to the one that has the maximum response:

$$l_t = l\left(\arg\max_{x \in X^d} p(y|x)\right) \tag{3.1}$$

This means at time t, the target is equally likely to be within a radius of d of the tracker location at time $t-1$:

$$p(l_t|l_{t-1}) = \begin{cases} 1, & \text{if } \|l_t - l_{t-1}\| < d \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

For updating the appearance model, the tracker creates two bags. One positive bag containing a set of patches $X^{pos} = \{x : ||l(x) - l_t|| < r\}$, where $r < d$. This bag will contain the positive samples for learning from the patches taken from the player. Every patch in this bag will represent some part of the player. The other bag will be a negative bag containing patches $X^{neg} = \{x : r < ||l(x) - l_t|| < \beta\}$ where $r < d$ and

β is another scalar. As one can see, there will be more negative samples than positives. To keep the training balanced, we subsample the negative bag randomly. This negative bag will represent the surroundings of the player.

When searching for the location of the object in a new frame, the tracker crops out image patches from the image. Once it finds the location with the maximum response, it updates the current state accordingly.

3.2 Multiple Instance Learning

Most learning algorithms require that their data is in the form $\{(x_1, y_1), ..., (x_n, y_n)\}$ where x_i is the instance of the data while y_i is a binary label for that instance. In our case the instance of the data is in the form of bags. So, the training data is of the form $\{(X_1, y_1), ..., (X_n, y_n)\}$ where $X_i = \{x_{i1}, ..., x_{im}\}$ is the bag, and bag label is defined by:

$$y_i = \max_j y_{ij} \tag{3.3}$$

where y_{ij} are the instance labels. So, if a bag contains even one positive instance, it is considered to be a positive bag. OMIL uses the MILBoost algorithm proposed by Viola et al. in [42]. This algorithm maximizes the log-likelihood of bags by using the gradient boosting framework:

$$L = \sum_i \log p(y_i|X_i)\big) \tag{3.4}$$

As we can see, we aim to train an instance classifier which will estimate $p(y|x)$. The labels of the instances are not known at the time of training. Thus, the likelihood is calculated for bags instead of the instances. The tracker uses the Noisy-OR (NOR) model to relate the probability of a bag being positive to its instances as follows:

$$p(y_i|X_i) = 1 - \prod_j \big(1 - p(y_i|x_{ij})\big) \tag{3.5}$$

Due to equation 3.5, if the probability of one of the instances in a bag is high the probability of the whole bag is high.

3.3 Online Boosting

The OMIL algorithm uses the MILBoost framework which combines a set of weak classifiers ($h(x)$) into a strong classifier ($H(x)$) as follows:

$$H(x) = \sum_{k=1}^{K} \alpha_k h_k(x) \tag{3.6}$$

where α_k are scalar weights assigned to each of the weak classifiers. These weights are adjusted after every iteration by assigning more weight to the misclassified instances. If the classifier is a decision stump, the feature having the most discriminating power is chosen for the entire training set. Thus the boosting performs feature selection by choosing a set of K features which is smaller than the feature pool, that have more discriminating power. Such feature selection proves advantageous for video tracking because the features are selected at runtime, creating the classifier that is optimal for the

current object in the video.

Typically boosting algorithms learn this type of model in batch mode. They train the weak classifiers sequentially and then after each weak classifier is trained, they update the weights based on misclassification. The OMIL algorithm learns the model in online mode. As we know that the instance weights and the classifier weights depend on the errors of the weak classifier, this algorithm keeps track of the individual weak classifier errors by keeping a running average of error for each h_k. Due to this error, the algorithm can update the weights of the classifier in an online fashion.

The entire data is not available at any given time for the model update. Since the algorithm is an online algorithm instead of the batch processing one, we cannot use decision stumps for selecting the most discriminating features. We have to preselect the features. To do so, we maintain a pool of $M > K$ weak classifiers. At the time of updating weights, all the M classifier weights are updated, but the algorithm chooses K sequential classifiers from this pool by keeping running averages of errors and updates the weights for h accordingly.

3.4 Online Multiple Instance Boosting

In boosting algorithms, the goal is to optimize a specific objective function J. So, weak classifiers are chosen to optimize the following criteria:

$$(h_k, \alpha_k) = \arg\max_{h \in \kappa, \alpha} J(H_{k-1} + \alpha h) \tag{3.7}$$

where κ is the set of weak classifiers and H is the strong classifier made up of $k-1$

weak classifiers in sequence. The classifier is updated to maximize the log-likelihood according to equation 3.4. The instance probability is given by computing the sigmoid of the strong classifier for that instance: $p(y|x) = \sigma(H(x))$, where σ is the sigmoid function. The bag probability is modeled by the NOR model given by equation 3.5. For simplicity the weak classifiers return real values than binary ones by absorbing the a_k. The algorithm keeps track of $M > K$ weak classifiers at all times. When the tracker updates the weights, it updates the weights of all of the weak classifiers in parallel. As seen before all the weak classifiers are instance classifiers and the tracker passes the label of the bag as the label of the instance to the weak classifier for training purposes. It chooses K weak classifiers from the classifier pool which maximizes the log-likelihood of the bags:

$$h_k = \underset{h \in h_1, \ldots, h_M}{\arg\max} \; L(H_{k-1} + h) \tag{3.8}$$

3.5 Discussion

Using Multiple Instance Learning to train the appearance classifier results in more robust tracking as shown by the experimental results on many challenging videos in [6]. These results show that OMIL tracker is robust with respect to partial occlusions and various appearance changes. Thus this tracker is efficient in addressing the challenges of camera motion and partial occlusion that we have stated in chapter 1.2. Since the tracker crops a set of image patches around the target object in the next frame, and searches for the target player in the image patches near the last location not taking into account any prior

motion it also addresses varying player movement.

Even then, this method cannot completely cope with challenges where object is completely occluded for a long period of time, or if the object leaves the frame completely. Like most adaptive appearance model based trackers, this tracker too will start learning from incorrect examples and lose track of the object. Note that the likelihood being optimized in Equation 3.4 is computed only on the current examples. Thus, it has the potential of over fitting to current examples, and not retaining information about previously seen data. To overcome these short comings of the tracker we have developed an interactive user interface.

Chapter 4: User Interface

The OMIL tracker worked well when the player is clearly visible throughout the video or if there is little or partial occlusion. We chose this tracker because it address the challenge of camera motion and partial occlusion. Since it uses an adaptive approach to tracking, it can also track players for a long time even if they change in appearance. It however does not address all of the challenges that have seen in section 1.2. In order to improve the output of the tracker we created a user interface (UI) by which the user will be able to correct errors that the trackers makes in real time. While the player is being tracked, the user can pause, fast-forward, rewind the video at any time and reselect the player position in any frame of the video. The system will accommodate this user defined position in the path of the tracker. To ensure smooth trajectories, we developed an interpolation strategy to always keep the path of the player smooth. The UI therefore, aids the tracker in finding the correct path for the player. The UI addresses the issues of complete occlusion and visual similarity between players which the tracker cannot handle, as it causes the tracker to become confused and track the wrong object.

In section 4.1, we explain the list of operations that a user can perform using the created UI and their effects. We then discuss about the interpolation strategy that will happen transparently as a result of the user interactions in section 4.2.

4.1 UI Overview

We developed the User Interface in such a way that the user will be able to correct the position of the player in any frame. Using the UI, the user can perform the following list of operations:

1. Open a video for tracking
2. Draw a bounding box around a player
3. Track multiple players on the same video
4. Rewind or Fast forward the video using track bar
5. Save the trajectory of all tracked players

These operations allow the user to give the necessary feedback to our system. The system uses this feedback to improve the accuracy of the tracker. It incorporates the feedback into the trajectory of the target player. To ensure smoothness of trajectory, the system alters the trajectory from the tracker by interpolating between the user-specified position and the last position estimated as correct. This last correct position is estimated by taking into account the confidence of the tracking for the frames in the previous two seconds of the video. The interpolation is threefold: trajectory, position and size. For each frame, the position and size of the bounding box is adjusted according to these three factors of interpolation. We discuss interpolation in detail in section 4.2.

As we can see in Fig 4.1, the UI is simple and easy to understand. It has a menu bar and additional buttons to help the user in providing the necessary feedback. These features of the UI help the user in the following manner:

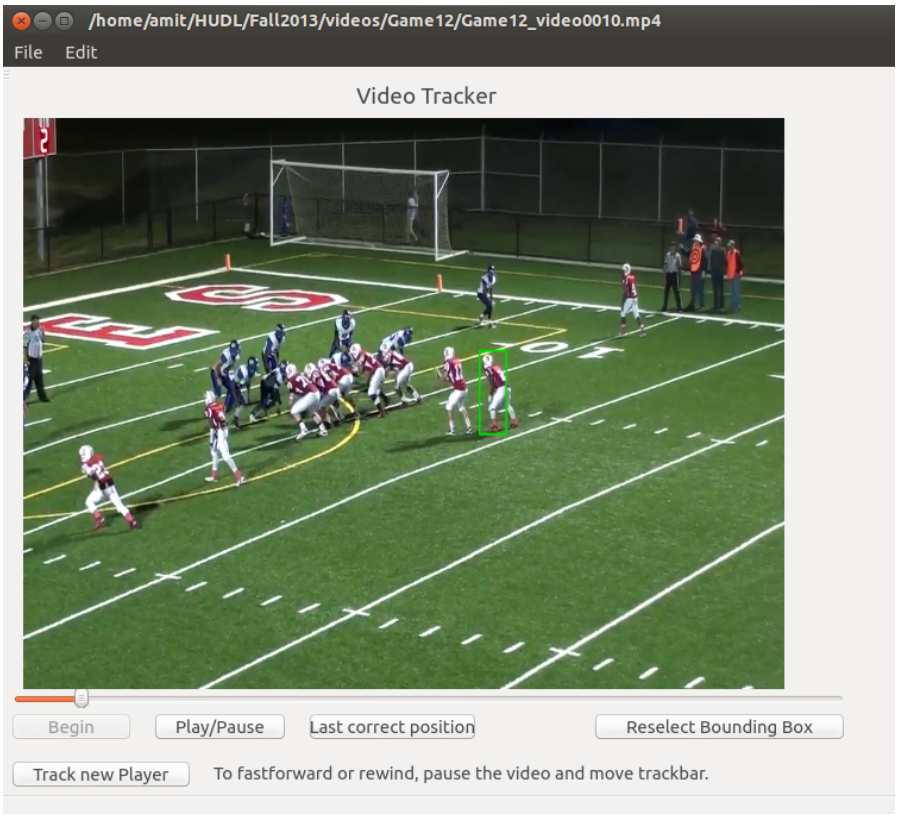

Figure 4.1: Interface : As seen in the figure, the target player has been marked by the bounding box. At the start of tracking, the user must press the 'Begin' button and specify the first bounding box for the target player. During tracking, the user can press the 'Pause/Play' button and the video will pause or play. The user can tell the correct position of the player by pressing the 'Reselect bounding box' button and draw a new bounding box on the target player. The user can choose to track a new player over the same video by pressing the 'Track new player' button. The user can also specify correct tracking positions by pressing the Last correct position' button. The user can save the trajectories of the players over the video by using the 'Save' command in the menu. The user can also undo the last feedback through the 'Edit/Undo' menu.

Menu/Open: The user an choose any video on disk and open it for tracking using this command.

Menu/Save: At any point during the tracking the user can opt to save the whole video with a bounding box around the players that were tracked up to that point. The system will ask the user for a location and name for the saved video.

Edit/Undo: If the user makes an error in the selection of the bounding box, we give him an alternative to undo the selection.

Begin: When the user opens a video from the menu, it will start playing. The user can initiate the tracking by pressing the 'Begin' button. This will open another window with the current frame and the user must select the player that he wants to track. This is done by simply holding down the mouse button and drawing a rectangular bounding box from top-left to bottom-right around the player. After this initialization, the tracker will start tracking the player.

Play/Pause: At any point during the video, the user can Play/Pause the video. After pausing the video, the user can also move the track bar to rewind the video. This enables the user to see the trajectory of the tracked player.

Last correct position: The user can click this button to notify to the system that the current frame is the last correct frame before the tracker has made a mistake. This is an optional feature in the system. The user can use this button to help the tracker to define the 'interpolation window' . We will talk about the interpolation window in detail in the next section.

Reselect Bounding box: On pressing this button the current frame will pop up in a new window. The user can then select the player by drawing a bounding box around the him. The tracker then uses this new position of the player as the current player position and starts tracking the player from the current position. The system will also interpolate

a few frames before the current position to keep the trajectory smooth. We will discuss about this interpolation in detail in chapter 4.2.

Track new Player: When the user has finished tracking one player, he can use this button to track another player in the same video. The trajectory of the first player will be saved and the user can track the second player in the same way the first one was tracked. In this way, the user can track as many players as needed one after another and save one video having the trajectories of different players at the same time.

Slowed Video: When the user will select a video, the video will start playing. The video is slowed down by a fraction of $1/6^{th}$ so that the user will have sufficient time to understand what is happening and make decisions accordingly.

Discussion

The UI is implemented in the above manner in order to help the user to provide feedback with minimal interaction. We made the UI easy to learn so that only a set of guidelines is sufficient for a user to successfully use our tool. No training is required. In the UI the target player is denoted by a bounding box. This is done because the tracker requires a bounding box around the target object. Also, the bounding box helps the user to accurately specify the area of the field in which the player is located. This would not have been possible if we had used a dot or a line to denote the user. In both these cases, the dimensions of the player would have to be guessed which would have had a negative effect on the tracking. If the user was denoted by a circular patch at the feet of the player, the tracker would definitely look nicer, but drawing a circular region would

be more complex for the user than drawing a rectangular bounding box for the user. Also, fast forward and rewind operations are easy to perform by just moving the track bar making the UI user-friendly.

4.2 Interpolation

The user can reselect the position of the player at any time during the video. There can be occlusions for a long time of the tracked player during the play in American football videos. Since the tracker uses an adaptive appearance model, it may learn the wrong object and track that object instead of the correct player. To address this issue, we have provided the user an opportunity to correct the tracker. The user can reselect the position and the tracker can get back on tracking the correct player. Whenever the user selects a new position for the player, we smoothen the trajectory of that player using our interpolation strategy.

The user is asked to reselect the player position if the tracker has gone wrong during the tracking of the player. So, we can make the assumption that the tracker has made an error in the frames leading up to the reselected frame. In order to correct the error, we first have to find the moment where the error occurred, say frame f_p. We call the time interval from f_p to f_{p+I} as the Interpolation Window I, where I is the number of frames in the window. To have knowledge about the trajectory after the user feedback, we allow the tracker to track the target object for 10 frames before interpolating.

As we can see in fig 4.2, we have two user defined frames f_1, f_{p+I}; and an interpolation window of length I frames from f_p to f_{p+I}. We interpolate the position of the

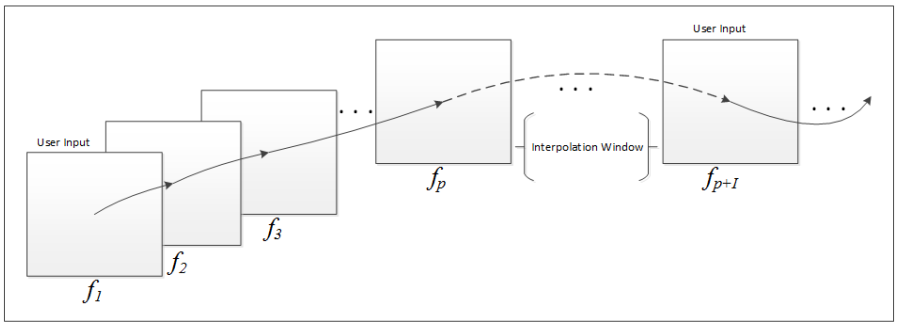

Figure 4.2: The Interpolation Window is shown in this figure. It represents the number of frames where the trajectory of the target player is corrected by the interpolation strategy.

player in all of the frames within this interpolation window to ensure smooth trajectory.

We get a confidence estimate as log-likelihood of the target player being in that position, as estimated by the tracker. We denote the confidence of the tracker as $C = \{c_i | i = 1, ..., p + I\}$, where C_i is the confidence of the tracker in frame f_i. We refer to the positions of the target player in each frame by the top left corner of the bounding box along with its height and width. We denote the position of the player as $\Omega = \{(x_i, y_i, w_i, h_i | i = 1, ..., p + I + 10)\}$, where (x_i, y_i) is the position, w_i is the width and h_i is the height of the bounding box in frame f_i. Algorithm 2 lays down the steps of the interpolation strategy.

Estimation of Interpolation Window: As we have seen, the interpolation window is the time interval during which the tracker is least confident about the target object, just before the user input. To calculate the interpolation window, we take into account the confidence estimate of the tracker. As we can see from chapter 3.4, we have real values returned by the tracker's weak classifier. We can use this value as a confidence estimate, as this value is the log-likelihood of the player being in the bounding box output by the

Algorithm 2 Interpolation Strategy

Input: f_1, f_{p+I}, C, Ω.
Output: $\hat{\Omega} = \{(x_i, y_i, w_i, h_i) | i = 1, ..., p + I\}$

1: Using algorithm 3, estimate Interpolation Window I.
2: $\Psi = \{(x_i, y_i) | i = 1, ..., p, p + I\}$
3: $\hat{\Psi}$ =spline interpolation(Ψ)
4: $\Omega_{before} = \{(x_i, y_i) | i = 1, ..., p\}$
5: $\Omega_{after} = \{(x_i, y_i) | i = p + 1, ..., p + I + 10\}$
6: A_{before} = acceleration(Ω_{before})
7: A_{after} = acceleration(Ω_{after})
8: Interpolate the acceleration $A_{IW} = \{(a_i) | i = p + 1, ...p + I\}$
9: $step_w = w_p - w_{p+I}/I$
10: $step_h = h_p - h_{p+I}/I$
11: $cur = x_p$, $size_w = w_p, size_h = h_p$)
12: **for** $i = f_p$ To f_{p+I} **do**
13: $cur = x_i + a_i$
14: $size_w = size_w + step_w, size_h = size_h + step_h$)
15: $\hat{\Omega}_i = \{\hat{\Psi}_k, size_w, size_h\}$, where $x_k = cur$
16: **end for**
17: **return** $\hat{\Omega}$

tracker. The higher the log-likelihood for a particular position, the higher the confidence that the player is in that position.

The first moment when the tracker is likely to make an error, the confidence of the tracker will be low. We have used this fact to estimate the moment when the tracker was least confident of the target player. We formulate this as a maximization problem where we try to cluster the set of confidence values of sequential frames into two groups with maximum variance between the two groups. This approach is explained in the algorithm 3.

We cluster the frames into two clusters. One cluster starts from f_1 while the other starts from f_{p+I}. The two clusters meet at a point that varies from 10 to 60 frames from

Algorithm 3 Estimation of Interpolation Window

Input: f_1, f_{p+I}, C, $\Omega = \{(x_i, y_i)|(x_1, y_1), ..., (x_{p+I}, y_{p+I}), ..., (x_{p+I+10}, y_{p+I+10})\}$.
Output: I

$\varepsilon = 0$
$I = 0$
for $i = 10$ to 60 step 5 **do**
    $Cluster_1 = \sum_{k=f_1}^{f_{p+I}-i} C_k$
    $Cluster_2 = \sum_{k=f_{p+I}-(i+1)}^{f_{p+I}} C_k$
    $\sigma =$ Median($Cluster_1$) $-$ Median($Cluster_2$)
    if $\varepsilon < \sigma$ **then**
        $\varepsilon = \sigma$
        $I = i$
    end if
end for
return I

f_{p+I}. The maximum variance between these two clusters denotes a drop in confidence values near f_{p+I}. This drop in confidence value states that there was significant change in the target object, which may be the cause for the error in tracking. Hence, we get the interpolation window for that instance of tracking.

Apart from this approach, the user can enter the last correct tracked position using the interface if the user feels that the estimate for the Interpolation Window has gone horribly wrong and the user defined position will override the Interpolation Window calculated using confidence. In this case, the system will use the Interpolation Window that the user has explicitly provided.

Trajectory Interpolation: Using the interpolation window, we can predict the trajectory that the player might have followed. We use the positions of the bounding

box in each frame excluding the frames that are in the Interpolation Window, i.e $\Psi = \{(x_i, y_i)|(x_1, y_1), ..., (x_p, y_p), (x_{p+I}, y_{p+I})\}$. We use spline interpolation technique to get $\hat{\Psi}$, that denotes the trajectory that the player follows during the interpolation window. $\hat{\Psi}$ has a value for y for every x from the least $x \in \Omega$ to the max $x \in \Omega$.

Positional Interpolation: Ψ gives us every location of the player along the trajectory of motion. To estimate the correct position for the player at a given frame i, we take into account the acceleration of the target player before and after the interpolation window, i.e. A_{before} and A_{after} respectively. We calculate this acceleration by only using the x co-ordinate of the location of the target player. Using these accelerations, we then interpolate the acceleration during the time of the interpolation window. This gives us acceleration for every frame during the interpolation window along x-axis and we can estimate the x-axis location of the player. Using this x-axis location and $\hat{\Psi}$, we get the location of the player along both axis. This location of the player is estimated in terms of location as well as time frame.

Size Interpolation: There may be the effect of camera motion during the interpolation window. To address this issue, we need to interpolate the size of the bounding box during the interpolation so that the it tightly fits the target player. There are several reasons why the size of the player may change during a play. For example, there is zooming during the play or the target player runs across the field. In all such cases, we can assume that the change in size of the player will be linear. So, for interpolation of the size of the bounding box for the player we have used a simple linear approach. We use the size of the bounding box for the last confident frame f_1 and the size of the bounding box for the latest user defined frame f_{p+I}. Assuming linear change in shape, we calculate

the step size for change in width as $step_w = width(f_1) - width(f_{p+I})/I$ and step size for change in height $step_h$ as $step_h = height(f_1) - height(f_{p+I})/I$. For each frame in the interpolation window, we increment the width and height of the bounding box by the step sizes to get the linear interpolation of the size of the bounding box.

Chapter 5: Experiments

5.1 Dataset

In our experiments we tracked a player in 129204 frames of video, in the 355 videos that we have used. These were selected by a web-service company that specializes in storing and providing American football videos to high schools. They selected 355 videos and suggested players to be tracked as typical cases from their database in an attempt to cover the wide diversity of video. We annotated the ground truth trajectories for each player selected in each video to allow for more refined evaluations.

We have different metrics for testing the different parts of the system that we have developed. We evaluate the system for accuracy, quantity of user interaction and analyze the effect of interpolation technique on the trajectory of the target player. We have tested the system on 355 American football videos. We have considered a variety of plays such as running plays, passing plays, kicking plays, etc. In each of the plays, we have tracked those players who have played an important role in that play. For example, we have tracked the guard if he makes a good tackle or we have tracked the wide receiver if he makes a good catch and scores a touchdown. We have categorized our results according to the different player types. This categorization is helpful in the analysis of the tool for a user, as the generated statistics are different for different player types. We have considered three different player types:

1. **Type 1 : Quarter Back / Running Back**

   All of the players that would run with the ball are categorized in this type.

2. **Type 2 : Wide Receivers / Corner Back**

   This type includes the players that are present on the sides of the play closer to the sidelines.

3. **Type 3 : Guards / Tackle / Linebacker**

   The defenders who make tackles or the guards who try to guard the player with the ball come under this category.

5.2 Metrics for Evaluation

We have tested the accuracy of our system and have found that the performance of the tracker increased with the addition of the user interface that we have developed. In order to test the accuracy of our system, we require ground truth for the tracked player for every video. We have manually created the ground truth for all of the videos that we have tested on. We denote the ground truth as a bounding box (location and size) in each frame of the video. We say that a frame f_i is 'accurate' if more than 65% tracker's bounding box overlaps with the ground truth bounding box. We calculate the accuracy as

$$Accuracy = \frac{\#accurate frames}{\#total frames} \times 100 \tag{5.1}$$

As you can see from Table 5.1, we have provided a boost in accuracy in the by including the user interface over the basic OMIL tracker. The increase in accuracy is more pro-

nounced in the case of videos of type 1 and type 3 as these videos have more challenging environment. The players in these videos are often occluded or near other players having similar appearance which confuses the tracker. The user feedback helps the tracker in such cases and the tracker has become more accurate.

Player Category	Number of videos	Accuracy without UI	Accuracy with UI
Type 1	120	48.30	89.70
Type 2	130	81.46	93.46
Type 3	105	56.16	81.32

Table 5.1: Accuracy of tracking with and without UI

In videos of type 1, the players typically get occluded at the start of the play. If the tracker looses track at such an early stage in the video, the tracker is accurate for a very small number of frames during the video giving rise to very low accuracy. If at this time, we correct the location of the player, it tracks the player for a longer time giving rise to higher accuracy.

In videos of type 2, the accuracy gain is not so much as typically the player is not occluded in this type of videos. So, the tracker performs well on its own and the additional UI causes increase in accuracy by a smaller margin.

In videos of type 3, the players are occluded several times in the video as they are usually present amongst other players. As a result the tracker performance is not that accurate. But with the inclusion of the UI, the tracker performs much better.

5.3 Analysis of User Interaction

The boost in accuracy due to the introduction of the UI comes at the price of user involvement in the tracking process. We have analyzed the number of user interactions required per play for each category. Table 5.2 shows the average number of times the user has to be involved in the tracking process per play category.

Player Category	Number of videos	Average Number of User Interactions per video
Type 1	120	1.35
Type 2	130	0.68
Type 3	105	2.25

Table 5.2: Number of User Interactions required

As you can see from table 5.2, the user needs to tell the correct position of the player more than once for the plays of Type 1 on average. The main reason for this is that in such plays the player is usually occluded at the very beginning and the tracker needs to be told of the position of the player for the tracker to get good accuracy. After this initial reset the player is not often occluded and more restarts are not required.

For plays of type 2, the least user involvement is required. The reason for this is that the players are often apart from the crowd and are not often occluded.

The plays of type 3 require the most user involvement. This is because the players in this type are often in the crowd and get occluded the most. On average such plays require user interaction more than two times in a play.

These results are the effect of training the user to maximize the accuracy with minimum number of user interactions. The user is given a set of guidelines which if adhered

to give maximum effectiveness of our system. These guidelines include that the user must reselect the position of the player after the player is completely visible. The user should ensure that the player is completely visible when the selection is done so that the tracker can learn the accurate appearance of the user. If this is done, the tracker will track the player successfully for the next frames and the interpolation method will try to keep the bounding box on the player as far as possible.

5.4 Analysis of Interpolation

The results in section 5.2 and 5.3 are calculated by considering an average user who would try to keep the bounding box on the target player for as long as possible by doing as little as possible. However, the accuracy of the system can be increased by providing more data to the system in the following manner.

For example, if the tracker is found to be going off the player in some part of the video, the user can go to an intermediate position and provide the system with an additional bounding box. This would call our interpolation method and correct the region where the tracker seemed to have gone wrong. A user in this way provide additional user feedback on each and every frame and achieve 100% accuracy for tracking. However it will require the user to manually select the position of the player in all of the frames, which is typically around 600 for an American football video. To provide the user with an estimate of the degree of accuracy to the number of restarts we have performed the following experiments.

In the videos that we have tested, we have calculated the accuracy statistics by lim-

iting the user interaction. We have noted accuracy of the tracking starting from no user interaction, till the time we get 100% accuracy. The graph presents the results of these experiments in figure 5.1.

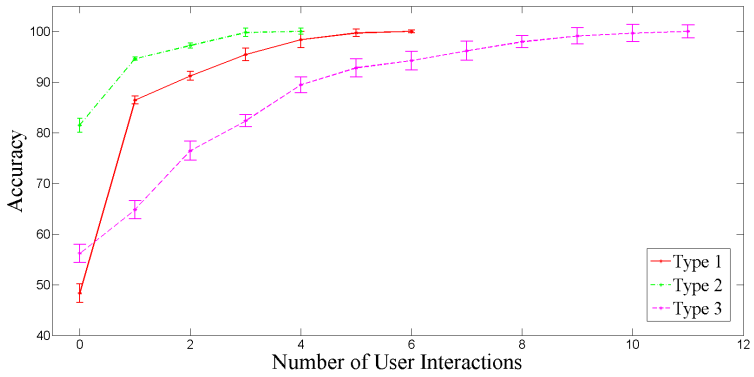

Figure 5.1: Comparison of the number of user interaction required for the three different player types. Note that here number of interaction is 0 means the use of original tracker.

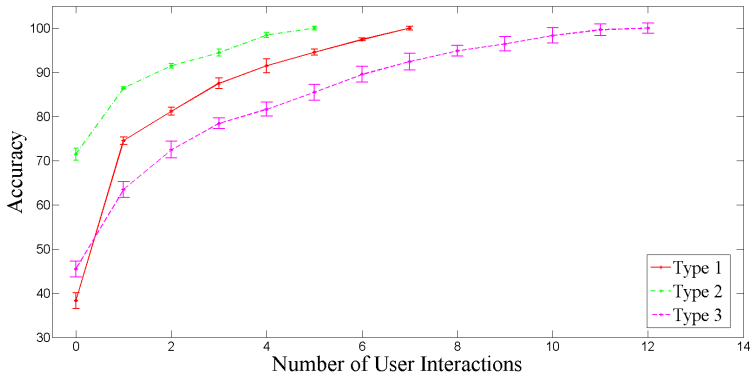

Figure 5.2: Comparison of the number of user interaction required for the three different player types using Particle Filter tracker. Note that here the number of user interactions is 0 means the use of original tracker.

The interpolation strategy that we have developed gives 100% accuracy for all three

categories when the user annotates sufficient amount of frames for the video. Since there is least occlusion in videos of Type 2, it requires lesser number of interactions to reach 100% accuracy. For Type 2, since the occlusion is usually at the beginning of the video, the accuracy is lower than that for Type 3 for no interactions. But after the first occlusion is corrected, the accuracy increases quickly. For Type 3 videos, the accuracy gradually increases as there are many occlusions during the entire course of the play. These videos also require many interactions to get to 100% accuracy.

Figure 5.2 shows the results of the implementation our interactive approach on a off-the-shelf implementation of the state-of-the-art particle filtering tracker [34]. This method performs poorer than OMIL tracker (when $m = 0$, where m = number of user interactions) agreeing with our review in chapter 2. But we can improve the accuracy of this tracker with the addition of our system. It takes longer to reach 100% accuracy using this tracker as the tracker is not suited for our focus domain. This goes to show that the system that we have developed is not tracker dependent and can be coupled with any tracker to increase accuracy of the tracking, though for our focus domain, it works best with OMIL tracker.

Chapter 6: Conclusion

In this work, we developed an interactive software tool for tracking players in American football videos. Our review of the state-of-the-art in tracking suggested that existing trackers are not sufficiently accurate and robust. We focused our work on developing an interactive software tool that could enhance the accuracy of existing trackers. We used OMIL tracker as we found it to be best suited to our work. We developed a user friendly tool that enabled the user to provide feedback to the tracker. Using this feedback, the tool corrects the trajectory estimate of the target player. We developed an interpolation strategy that interpolates the trajectory, acceleration and size of the target player to ensure smoothness. Our experimental results demonstrate the boost in performance that the OMIL tracker receives when our interactive system is added. The accuracy of the tracker increased from 62% to 88% on average with the addition of our interactive tool to the tracker. We evaluated the user-friendliness of the tool by counting the average number of user interactions required per video in each category of videos. While this count depends on the video content, we observed a relatively small number of interactions necessary to achieve high accuracy over the variety of videos that we tested our system on. We also demonstrated that the tool is tracker independent by combining it with Particle Filtering, though it works best when used with OMIL tracker for our focus domain.

We believe our tool's main failing lies in the short-sightedness of our tracker. The

tracker uses boosting approach which is prone to over fitting to current samples. It does not take into account the historical data, which would allow it to overcome more lengthier occlusions. Another failing in our approach is the calculation of the Interpolation Window in chapter 4.2. This is done by considering the log-likelihood returned by the tracker as a confidence estimate. We believe that in cases where the tracker learns the wrong player, the confidence returned drops only for a very short period of time and is difficult to find accurately. In order to work around this case, we provide the user with the option of providing the tracker with the last correct frame. When the user gives the tool this information, it can compute the interpolation window without the need of a confidence estimate. A better notion of confidence would help the interpolation method in such cases and it would bring the number of user interactions required to a lower number.

American football videos present a rich variety of problems for tracking methods. We provide a tracking tool to facilitate solving various problems in tracking for future studies and research.

Bibliography

[1] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805. IEEE, 2006.

[2] Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. Keyframe-based tracking for rotoscoping and animation. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 584–591. ACM, 2004.

[3] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.

[4] Shai Avidan. Support vector tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):1064–1072, 2004.

[5] Shai Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261–271, 2007.

[6] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, 2011.

[7] Alexandru O Balan and Michael J Black. An adaptive appearance model approach for model-based articulated object tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 758–765. IEEE, 2006.

[8] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(9):1806–1819, 2011.

[9] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[10] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522. IEEE, 2009.

[11] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1273–1280. IEEE, 2011.

[12] Aeron Buchanan and Andrew Fitzgibbon. Interactive feature tracking using kd trees and dynamic programming. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 626–633. IEEE, 2006.

[13] Pushpender Prasad Chaturvedi, Amit Singh Rajput, and Aabha Jain. Video object tracking based on automatic background segmentation and updating using rbf neural network. *International Journal*, 2013.

[14] Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics, 2004.

[15] Robert T Collins, Yanxi Liu, and Marius Leordeanu. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1631–1643, 2005.

[16] Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM, 2005.

[17] Piotr Dollár, Zhuowen Tu, Hai Tao, and Serge Belongie. Feature mining for image classification. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[18] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via online boosting. In *BMVC*, volume 1, page 6, 2006.

[19] Helmut Grabner, Jiri Matas, Luc Van Gool, and Philippe Cattin. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1285–1292. IEEE, 2010.

[20] Gregory D Hager and Peter N Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(10):1025–1039, 1998.

[21] Weiming Hu, Xi Li, Wenhan Luo, Xiaoqin Zhang, Stephen Maybank, and Zhongfei Zhang. Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(12):2420–2440, 2012.

[22] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *Computer Vision–ECCV 2008*, pages 788–801. Springer, 2008.

[23] Rahul Jain, K Pramod Sankar, and CV Jawahar. Interpolation based tracking for fast object detection in videos. In *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2011 Third National Conference on*, pages 102–105. IEEE, 2011.

[24] Allan D Jepson, David J Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003.

[25] Michal Juza, Karel Marik, Jiri Rojicek, and Petr Stluka. 3d template-based single camera multiple object tracking. In *Computer Vision Winter Workshop*, 2006.

[26] Mark Keck, Luis Galup, and Chris Stauffer. Real-time tracking of low-resolution vehicles for wide-area persistent surveillance. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 441–448. IEEE, 2013.

[27] Minyoung Kim, Sanjiv Kumar, and Henry A Rowley. Object tracking in video with visual constraints, July 2 2013. US Patent 8,477,998.

[28] Benson Limketkai, Dieter Fox, and Lin Liao. Crf-filters: Discriminative particle filters for sequential state estimation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3142–3147. IEEE, 2007.

[29] Ruei-Sung Lin, David A Ross, Jongwoo Lim, and Ming-Hsuan Yang. Adaptive discriminative generative model and its applications. In *NIPS*, volume 17, pages 801–808, 2004.

[30] Xiaoming Liu and Ting Yu. Gradient feature selection for online boosting. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[31] Iain Matthews, Takahiro Ishikawa, Simon Baker, et al. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815, 2004.

[32] Greg Mori and Jitendra Malik. Recovering 3d human body configurations using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1052–1062, 2006.

[33] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. Object tracking with an adaptive color-based particle filter. In *Pattern Recognition*, pages 353–360. Springer, 2002.

[34] Katja Nummiaro, Esther Koller-Meier, Luc Van Gool, et al. A color-based particle filter. In *First International Workshop on Generative-Model-Based Vision*, volume 2002, page 01. Citeseer, 2002.

[35] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *Computer Vision-ECCV 2004*, pages 28–39. Springer, 2004.

[36] Jesus Pestana, Jose Luis Sanchez-Lopez, Pascual Campoy, and Srikanth Saripalli. Vision based gps-denied object tracking and following for unmanned aerial vehicles. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6. IEEE, 2013.

[37] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[38] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[39] Jian Sun, Weiwei Zhang, Xiaoou Tang, and Heung-Yeung Shum. Bidirectional tracking using trajectory segment analysis. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 717–724. IEEE, 2005.

[40] Sudheendra Vijayanarasimhan and Kristen Grauman. Active frame selection for label propagation in videos. In *Computer Vision–ECCV 2012*, pages 496–509. Springer, 2012.

[41] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[42] Paul Viola, John Platt, Cha Zhang, et al. Multiple instance boosting for object detection. In *NIPS*, volume 2, page 5, 2005.

[43] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

[44] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *Image Processing, IEEE Transactions on*, 22(1):314–325, 2013.

[45] Jianyu Wang, Xilin Chen, and Wen Gao. Online selecting discriminative tracking features using particle filter. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1037–1042. IEEE, 2005.

[46] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418. IEEE, 2013.

[47] Yuehua Xu and Alan Fern. On learning linear ranking functions for beam search. In *Proceedings of the 24th international conference on Machine learning*, pages 1047–1054. ACM, 2007.

[48] Ting Yu and Ying Wu. Decentralized multiple target tracking using netted collaborative autonomous trackers. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 939–946. IEEE, 2005.

[49] Liang Yuan, Yuan F Zheng, Junda Zhu, Lina Wang, and Anthony Brown. Object tracking with particle filtering in fluorescence microscopy images: Application to the motion of neurofilaments in axons. *Medical Imaging, IEEE Transactions on*, 31(1):117–130, 2012.

[50] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparsity-based collaborative model. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1838–1845. IEEE, 2012.