# Converting 2D motion into 3D world coordinates in the case of soccer players video

by Charalampaki Eirini
Technological Educational Institute of Crete, Department of Informatics Engineering, School of Applied Technology, 2017

THESIS

submitted in partial fulfillment of the requirements for the degree
MASTER OF SCIENCE

Approved by:
Major Professor

Dr. Athanasios G. Malamos
Technological Educational Institute of Crete
Department of Informatics Engineering

# Abstract

One of many important tasks in computer vision applications is real time object detection and tracking. Motion information and visual features, such as shape, color and texture, can be used to extract information from a video, to track and detect objects. Although classical tracking algorithms work accurately in perfectly arranged conditions, tracking objects in real-life situations poses challenges due to the multitude of variable conditions. However, efficient object tracking is a difficult task due to the multitude of the conditions met. Significant problems arise in computer vision systems, especially under the presence of noise, scene illumination, object shape variation and total or partial occlusion. The purpose of this thesis is to develop a method for effective object tracking and detection in complex scenes, such as sports game. In sports, the constant movement and the unexpected change of the velocity, is a common thing. Furthermore, sport players can frequently leave or enter cameras field of view, but also occlude each other. A robust solution could combine temporal motion information and statistical analysis of the visual features. The proposed approach would enable to accurately track targets in motion in real scenes.

# Contents

# List of Figures

# Chapter 1

# Introduction

Vision is the main form to achieve outside information for being [1]. According to scientific studies, it is found that the human retina can transmit data at roughly 10 million bits per second. By comparison, an Ethernet can transmit information between computers at speeds of 10 to 100 million bits per second [2]. Nowadays, computer vision, pattern recognition, computer graphics, artificial intelligence etc., are developed rapidly, making computers recognize, analyze, understand things as human. Based on object detection, visual tracking is presented which is applied to detect, extract, recognize and track moving objects in images sequence [3]. Moving parameters, such as position, speed, acceleration is processed and analyzed to understand behavior of moving target. Visual tracking is applied in multi-domains such as medical diagnosis, robots, video compression, human computer interaction, education and entertainment combining technologies of image processing, pattern recognition, artificial intelligence. In sports, especially soccer, computer vision systems are gaining tremendous growth due to its numerous uses for sport professionals, trainers and sport game viewers. Soccer video analysis has a wide range of applications, both at a group and individual level. Football players can gain insight into their physiological performance [4], for instance players' covered distance extraction and trajectory. Coaches can extract information about the quality of tactical analysis and game strategy, strengths or weaknesses evaluation of the team's opponents or player and moreover, the verification of referee decisions. Furthermore, players' tracking information can be used by the broadcaster of a football match to provide enhanced replays and statistical analysis for the footballs team supporter. Tracking football players effectively is a challenging task- due to the constant movement and the unexpected change of velocity. Moreover, athletes can frequently leave or enter a cameras field of view, but also occlude each other. The presence of noise and scene illumination lead to significant problems, when a classical tracking approach is chosen.

## 1.1 Objective

The aim of this thesis is to put to the test if it is achievable to implement an application for tracking single and several players on a football field. Considering the large area of the field, several cameras required to be used in a real system. In this thesis, a limitation is set to only cover a part of the football field. The camera that will be used is standard mobile camera.

## 1.2 Goal and Motivation

Classical tracking approaches work accurately under perfectly arranged conditions. On the other hand, novel tracking algorithms depend upon multiple cameras to extract as many motion information as they can of the object that is tracked. This thesis motivation, is the craving of a straightforward application of tracking football players, without the use of multiple cameras. Our goal is to overcome the challenges of tracking in a complex scene- such as a football match- and implement a system that tracks one or more football players effectively, with the convention that a single mobile camera is going to be utilized. The following goals have been defined for this thesis.

- A single mobile device is used, which covers part of the field.
- Track one player/ multiple football players.
- Real world coordinates estimation.
- Motion statistics extraction from real world coordinates.

## 1.3 Limitations

The following limitations has been set for this thesis: According to the initial scenario the software will run to a mobile device and thus only one camera will be available.

## 1.4 Applications

The proposed system, has numerous applications. Broadcaster companies will be able to monitor automatically the football game. Valuable information can be collected, since every player could be tracked, such as average speed, total running distance, maximum speed, ball's possession time. Furthermore, it can be a useful tool for coaches to gather information about the quality of tactical analysis and game statistics, post-game analysis, strengths or weaknesses evaluation of opponents team or player and moreover, the verification of referee decisions.

# Chapter 2

# State of the art

## 2.1 What is Object Tracking?

Object tracking is the process of locating one or multiple moving objects in successive frames of a video, utilizing a camera. In computer vision and machine learning, object tracking is a wide-ranging term that incorporates conceptually related but technically dissimilar ideas, for example, Kalman filtering- one of the most popular signal processing algorithm- is used to predict the location of a moving object based on prior motion information. Also, Dense optical flow algorithms estimate the motion vector of every pixel in a video frame and Sparse optical flow algorithms, track the location of a few feature points in an image, like the Kanade-Lucas-Tomasi (KLT) feature tracker. Moreover, there are two classes of trackers, single object trackers and multiple objects trackers. In single object trackers- to specify the location of the object we want to track- a rectangle is used to mark the first frame. Then, using the tracking algorithm, the object is tracked in subsequent frames. In multiple objects trackers, multiple objects are detected in each frame and to identify which rectangle in one frame corresponds to a rectangle in the next frame, a track finding algorithm should be used. Two very popular techniques that are also used for tracking, are Meanshift and Camshift algorithms for locating the maxima of a density function.

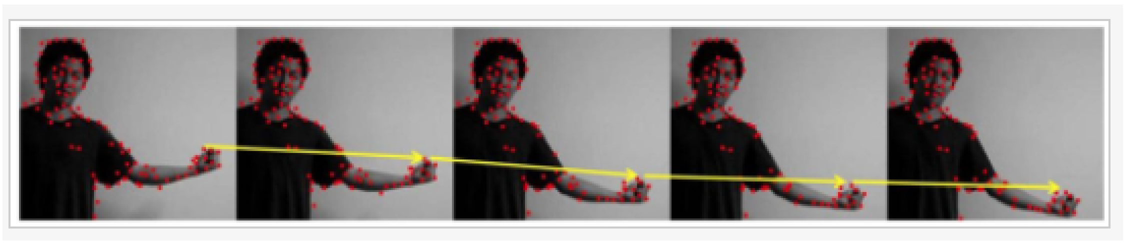


Figure 2.1: Example of fist tracking using one Lucas-Kanade-Tomasi(LKT) tracker [5]

## 2.2 Tracking versus Detection

A critical question arises. Why do we need tracking in the first place and not just do repeated detections? The answer is clear, detection algorithms are slower than tracking algorithms. When a detected object in the previous frame is tracked, many information about object's appearance are obtained. Furthermore, the location, the speed of its motion and the direction in the previous frame, are known. In the consequent frame, all this information can be used to predict the location of the object in the next frame. To locate the object more accurately, a limited search needs to be done around the expected location. A detection algorithm starts from the beginning, while a reliable tracking algorithm takes advantage of the acquired information about the object prior to that point. Accordingly, if we want to design an efficient system, most of the time, an object detection is employed on every $n^{th}$ frame, while we run a tracking algorithm in the n-1 frames in between. Someone may wonder, if it is more feasible to simply detect the object in the first frame and track afterwards. Tracking can take advantage of the information that are obtained, but the accuracy of the procedure is uncertain when the moving object goes behind an obstacle for a long period of time or moves too fast. Moreover, tracking algorithms often accumulate errors and the bounding box that tracks the object slowly slides away from the tracked object.

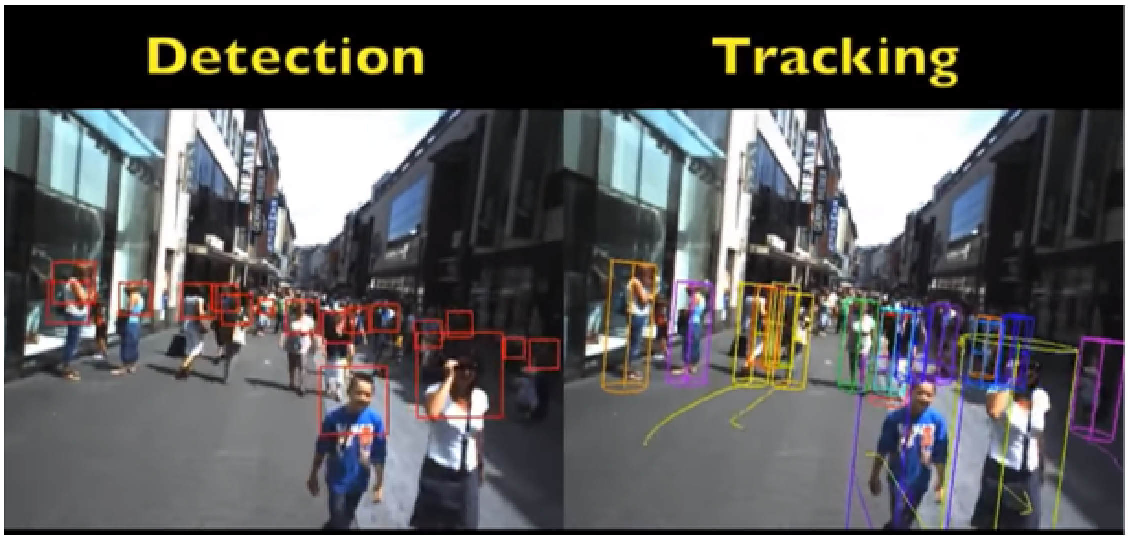


Figure 2.2: Tracking versus detection [6]

To deal with these problems, a detection algorithm is runned from time to time. Since detection algorithms are trained on object's instances, more information are known about object's general class. In a different manner, tracking algorithms have more knowledge about the specific instance of the class- they are tracking. An efficient tracking algorithm, will be able to handle occlusion, while detection fails. Furthermore, tracking algorithms help preserving object's identity. An array of rectangles that contain the object, is the output of object detection, yet there is no identity attached to the object. For example, if we want to detect 10 moving circles in a video- the output will be the rectangles matching to all the circles- the detector has detected in one frame. In the next frame, another array of rectangles will be the output. The problem is that in the first frame, a specific circle might be defined by the rectangle at location 12 in the array and in the next frame, it could be at location 15. As long as detection is being used on one frame, there is no indication which rectangle corresponds to which object, and the solution is tracking, due to the fact that provides a way to actually associate the circles.

## 2.3 Related work

Since the proccess of tracking objects is a difficult task due to the variety of the conditions met, different approaches have been proposed. In [7] authors introduced a new method of multi- object tracking by using multi-camera network, tested on 3 different datasets. They used single camera tracking and multi camera collaborative tracking. For single camera tracking, codebook background modeling algorithm is used and then in order to achieve more accurate detection, histogram of oriented gradient algorithm is applied. The results from each single camera used to propose- a multi camera view object correspondence scheme- to track objects in multiple camera views. Their proposed method can track objects effectively, even when they are occluded in some cameras and capture the differences in object's appearance by multiple views. Authors in Robust tracking of multiple persons in real-time video [8], combine object and background segmentation and especially temporal differencing to detect motion in the region of interest. As tracking strategy, they applied particle swarm optimization algorithm to overcome the robustness problem in noisy background. Their approach works faster and more precisely than the conventional particle filter.
Some researches for distinguishing the athletes from the background, relied on color of pants and shirt. Color-based classification techniques can be divided into template-based player detection and pixel-based detection. In the template based player detection approaches, each window in the image is classified into player or non-player[9]. On the other hand, each pixel is classified by its color component in pixel-based player detection methods [10]. The main disadvantage of these methods is that player's image might get fragmented into multiple regions.

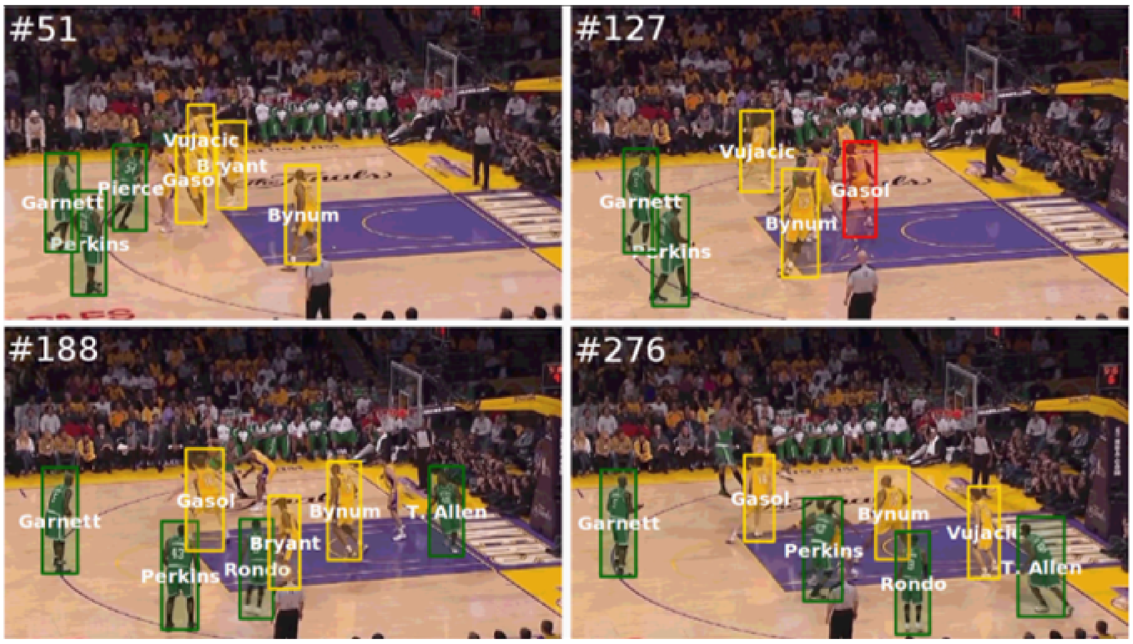


Figure 2.3: Learning to track and identify players from broadcast sports videos [16]

In [11] authors combined players' features based particle filter, number detection likelihood model and a motion vector prediction model for tracking multiple players in 3D space. They map 2D images to 3D space in the real world, by using multiple fixed cameras, which allow them to track players in 3D space. To distinguish players with different jersey number, they use the number detection likelihood model and to predict players' position when occlusion occurred, they use a motion vector. Authors in[12], proposed an automatic player detection by combining boosting detection and background modeling, unsupervised labeling and efficient player tracking method using Markov Chain Monte Carlo data association applied for broadcast soccer videos. However, whenever the video is blurred, camera is moving suddenly and unexpectedly or long occlusions occurred, their proposed method leads to failure. In the literature, in videos captured by static cameras, background subtraction method has been extensively used for segmenting moving players based on motion information[13] [14]. Nevertheless, in case of moving camera, the accuracy of this method is ambiguous.

In this paper [15], a blob-guided Particle Swarm Optimization is introduced for efficiently searching multiple players in an image. Authors- based on the number of blobs- divided the swarm into sub-swarms and search for players in each blob. The blob-guided Particle Swarm Optimization detects effectively multiple players, but several open issues need to be investigated. In literature, the discriminative color of the players' uniforms has been widely used, although there are unreliable results using just color information, to distinguish blurred and distant players. Moreover, if the players' uniform colors are similar color with the playfield, advertising billboards and lines, the detection results are uncertain.

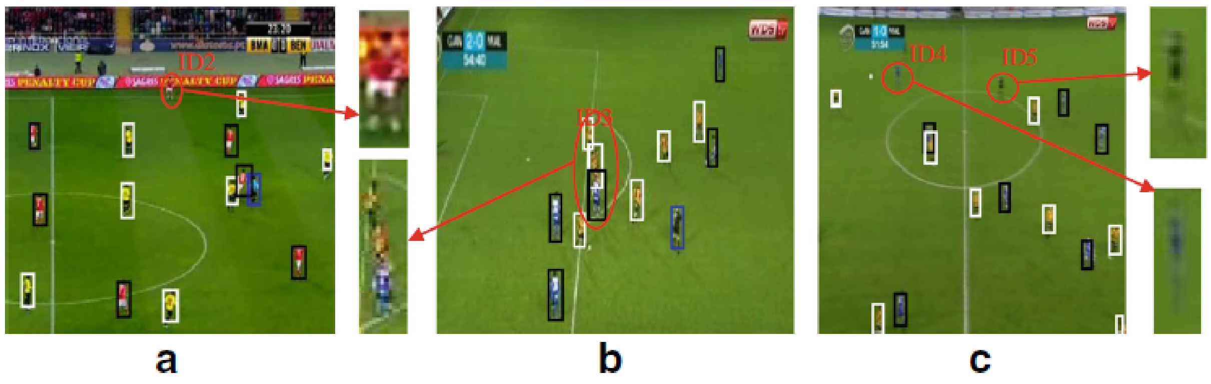


Figure 2.4: Examples of failure cases (a) Player merged with advertising billboard in similar color (b) Heavy occlusion (c) Blurriness [15]

# Chapter 3

# Methodology

## 3.1 The Football field

For our implementation, all video sequences are recorded in a live football match, taking place in a mini 5×5 football filed with 30 × 60 meters size. The lines have 8 cm width. The longer sides are called touchlines. The other opposing sides are called the goal lines. Goals are placed at the center of each goal-line. These consist of two upright posts placed equidistant from the corner flag posts, joined at the top by a horizontal crossbar. The inner edges of the posts must be 3 m apart, and the lower edge of the crossbar must be 2 m above the ground.

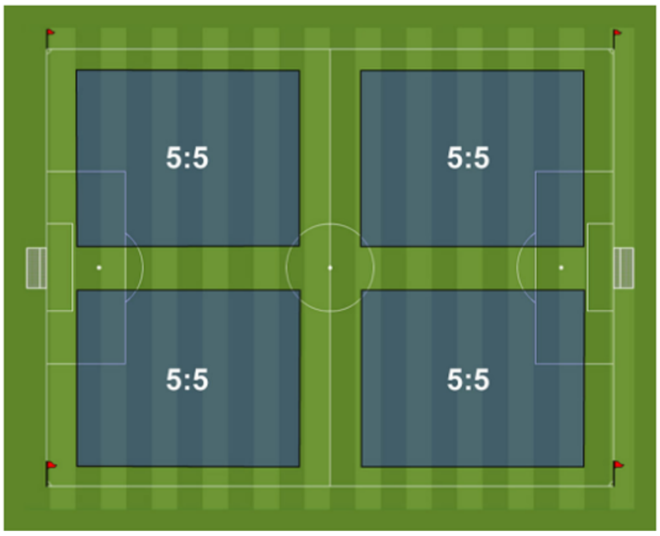


Figure 3.1: A mini $5 \times 5$ football pitch [17]

## 3.2 Camera setup

There are numerous ways to allocate the camera, all relying upon what is the chosen focal length, how many camera rigs are going to be used and how much overlap is preferable. The total cameras that can be used, the position of the camera, the camera's distance from the football field, camera coverage, whether the player should be observed from more than one direction, the football player's range resolution and what should be the player's pixel resolution at a specific distance, are significant factors that should be considered depending on the desired output. An example a possible setup- with eight cameras placed to cover a football field- is depicted below.

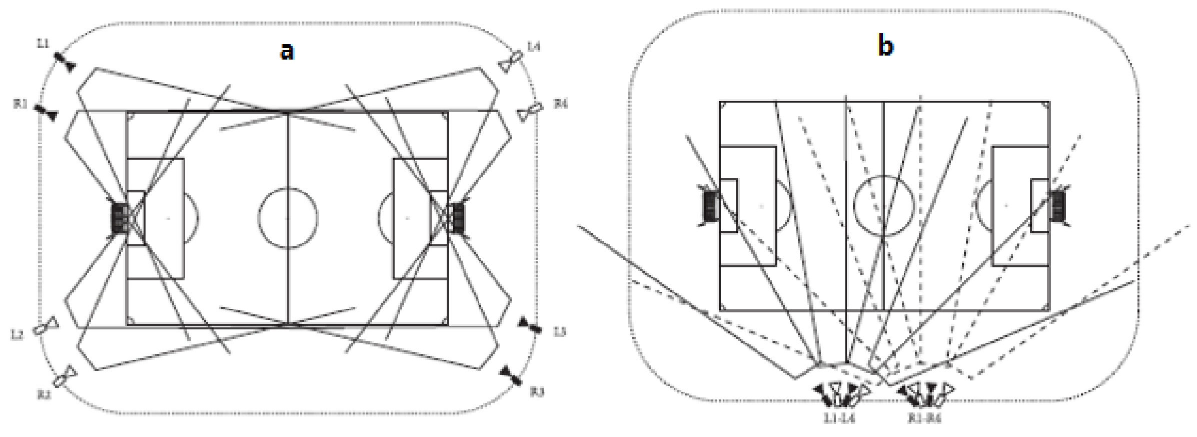


Figure 3.2: A way to cover the football field from two different corners
Figure 3.2-a: A way to cover the football field from two different corners. With this approach, football players are covered from two different directions, resulting in the improvement of the position's estimation in the current area. However, positioning the cameras this way, reduces the resolution in the middle of the football area, because of a wider FOV requirement. Figure 3.2-b: The division of the field into four pieces.

The number of the cameras to be used is for the most part an economical decision, not for the cameras' cost itself, but the rigs, the cables, etc. that could significantly increase the total price. The usage of less than eight cameras has shown that causes critical problems, since very wide-angle lenses have to be used, otherwise the resolution would be unsatisfactory. Player's resolution refers to the actual pixel resolution and the depth resolution. The actual pixel resolution is the size of the player at a specific distance, in image pixels. Depth resolution depends on the focal length, the camera base and the distance from the camera and measures the estimation of the distance between the player in a scene and the camera.

For our system, a steady mobile camera is selected, which only covers a part of the football field. The resolution of the video sequence is high definition 1280x720 size in pixels and the frame rate is 30 frames per second.

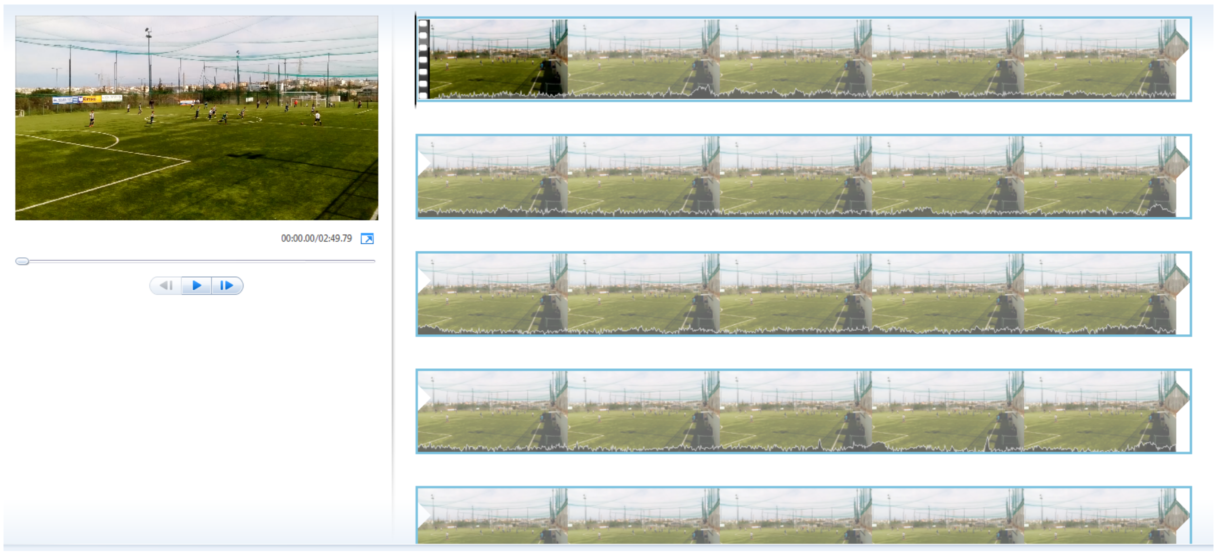


Figure 3.3: The chosen video sequence

## 3.3 OpenCV 3.2

For this thesis implementation, the C++ interface of open source library OpenCV 3.2 (Open Source Computer Vision Library) is used inside the Microsoft Visual Studio 2015. OpenCV [19] package includes a considerable amount of static and shared libraries.

The following modules are included in our application:

- Core functionality
- Video I/O
- Image Processing
- High GUI

OpenCV header is included in our application, which is a file that defines what modules where included during the build of OpenCV. All the OpenCV classes and functions are placed into the cv namespace. Therefore, to access this functionality from our code, we use directive namespace cv;. Moreover, additional libraries need to be included to our implementation the so called extra modules for OpenCV[19]. From opencv_contrib modules, roi selector header is included in our application, which is a class where the drawing mode is set. It notifies the user and prompt him to select an object- calling MouseCallback function- which sets the mouse handler for the specified window. When the user selects the preferable area, a bounding box is appearing with a crosshair in the middle of it. The drawing process is started from the center of the selected object and the selection process is finished, whenever the user is pressing the escape button from the keyboard. Region of interest selector is part of the long-term optical tracking API in OpenCV. "Long-term optical tracking is one of most important issue for many computer vision applications in real world scenario. The development in this area is very fragmented and this API is a unique interface useful for plug several algorithms and compare them. These algorithms start from a bounding box of the target and with their internal representation they avoid the drift during the tracking" authors in[20] pointed out. The tracking API of OpenCV includes six tracking algorithms: KCF, MIL-these two algorithms will be used in our implementation- Boosting, TLD, GoTurn and MedianFlow.

# Chapter 4

# Implementation

## 4.1 Track a single football player

In this implementation, firstly the input video sequence is specified as parameter of the application. Several variables need to be declared. The region of interest (roi) parameter is necessary to record the bounding box of the tracked player and this variables stored value will be updated in every iteration. Moreover, a frame variable is required to be set- to hold the image data- from each frame of the input video. Then, a tracker is created by its name and the selected algorithms to be used are MIL and KCF. A more comprehensive analysis for the selected trackers is taken place in the next section. After the creation of the tracker object and the setup of the input video, the user selects the object to be tracked: by using the roiSelector function, the user will be able to select a bounding box of the preferable object to be tracked using a GUI. The selection is created from the center of the bounding box and a middle cross will be drawn. Afterwards, the selected tracker is initialized for the first frame of the video input and the bounding box that surrounds the target is drawn.

At that point, the variables to store the motion extraction results- such as the distance, the selected object's center, the previous $x$ position of the object and the previous $y$ position of the object- are initialized. The previous $x$ and $y$ position of

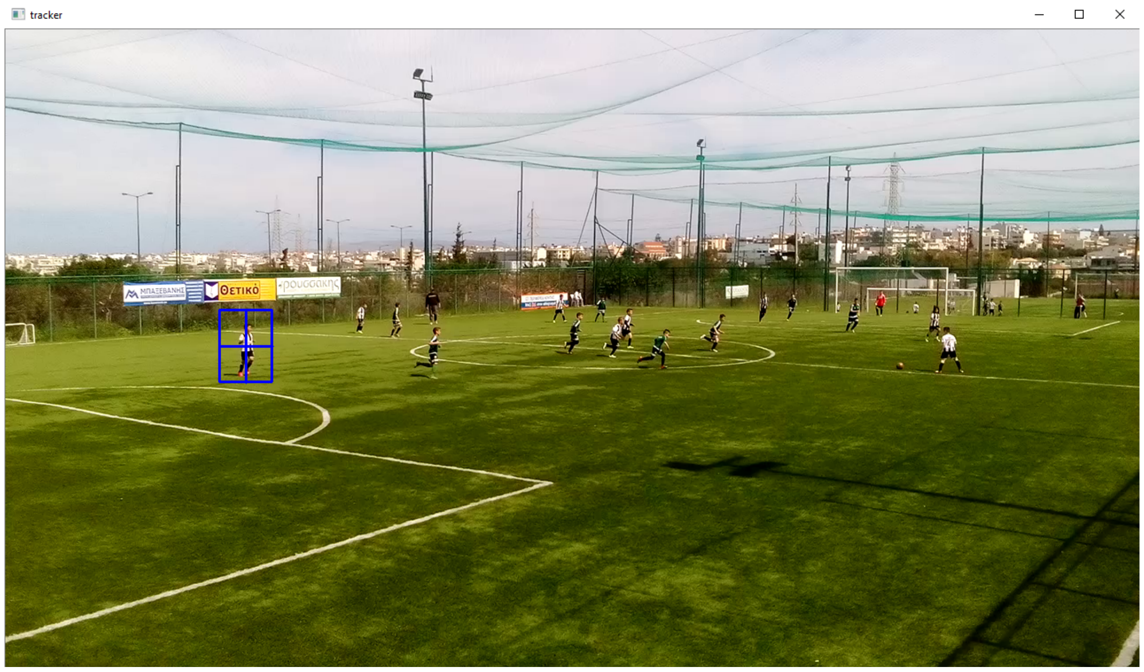


Figure 4.1: The selection of a single player

the object to be tracked, is assigned to the original $x$ and $y$ variable respectively. The tracking begins within an iterative process from the first frame of the video sequence until the last one. For finding the new most likely bounding box for the selected player, the tracker is being updated and the tracked object is drawn on the GUI window. At that point, the tracked object's distance is calculated, by using the Euclidean distance formula. The Euclidean distance between points a and b is the length of the line segment connecting them $\bar{ab}$.

In the cartesian coordinate system, if $a = (a_1, a_2, a_3, ., a_n)$ and $b = (b_1, b_2, b_3, ., b_n)$ are two points in Euclidean n-space, then the distance d from a to b, or from b to a is given by the Pythagorean formula

$$d(a,b) = d(b,a) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + .... + (a_n - b_n)^2} \tag{1}$$

The position of a point in a Euclidean n-space is a Euclidean vector. So, a and b are Euclidean vectors, starting from the origin of the space, and their tips indicate

two points. The distance of the tracked object, its original position and its position after the tracking process, are extracted in an output text file. Finally, the results of the image with the tracked object are eventually displayed on screen.
In figure 4-2 below, the frame number on the current frame is obtained and written in the top left of our GUI screen, and the selected football player is tracked throughout the whole video sequence.

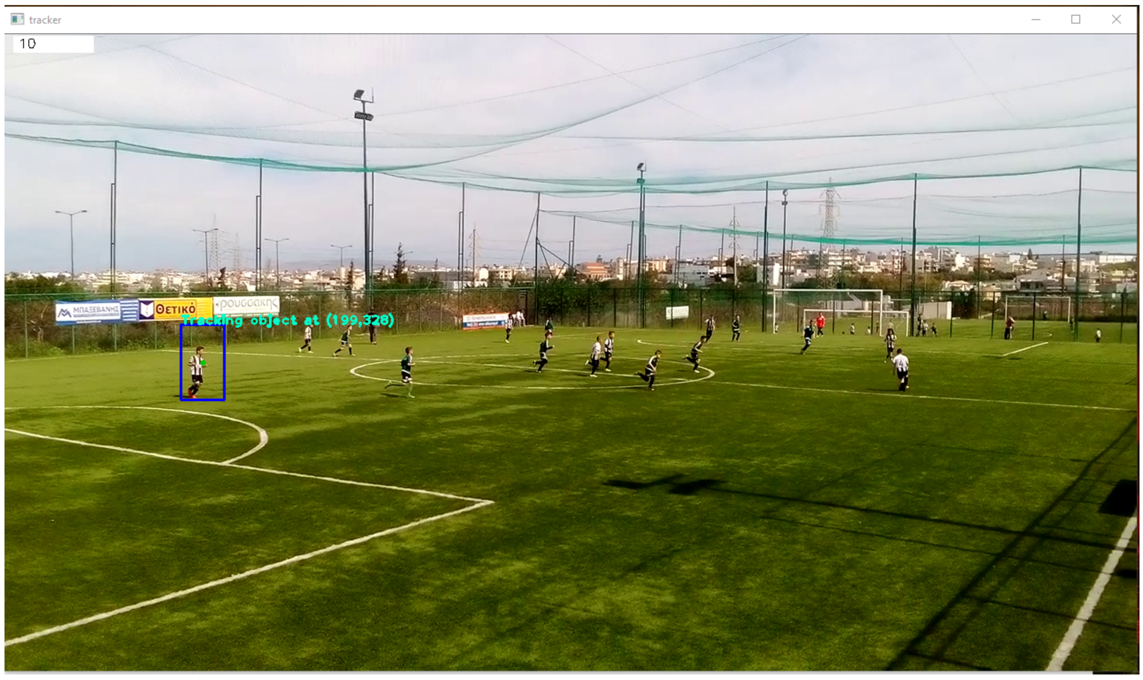


Figure 4.2: A single player tracking process

Furthermore, the center of the tracked player, as well as his current position are being displayed on the screen.

## 4.2 Track multiple football players

Here, we create a multi tracker object instead of a single tracker object, which has been created in the previous implementation. The difference in this approach, is that a container object of the multiple tracked objects should be defined. Moreover, a variable where the center of each selected player will be stored, requires to be set. In our proposal, the multi tracker object, uses the same tracking algorithm for all the objects to be tracked. For each tracked player, there is, though, the option of using different type of tracking algorithm. To achieve this, the tracking algorithm should be defined, whenever an extra object is added to the multi tracker object, however this option is somewhat out of the scope for this thesis.
After the creation of the multi tracker object and the setup of the input video sequence, the user is able to select multiple football players to be tracked. Once again, the region of interest function is used, to select multiple football players- with the result saved in the container object- that is mentioned above. Afterwards, the tracker is initialized for the first frame of the video and an infinite iterative process signifies the starting point of the tracking procedure throughout the whole video sequence. At this point, the tracking result is being updated, nevertheless a second difference from the single object approach is observed.
In our proposal, the multi tracker object- uses the same tracking algorithm for all the tracked objects and the user is able to select multiple football players. Considering the fact that the purpose of this approach is to track multiple football players, the drawing process is slightly different. A second iterative process is started, drawing the total number of the tracked multiple players. The range of this iterative process, starts from 0 to the number of football players, that the user previously selected. Once more, a green circle is defining the center of each tracked object and the motion information as well the position of each tracked player are stored in a text file. To calculate the processing speed, we start a timer that counts the number of clock ticks since our application started. As soon as our tracker is updated, we subtract the

number of clock ticks from that counter and divide the clocks per sec with that timer to calculate the processing speed. The processing speed can be seen in the top left corner of the GUI window.

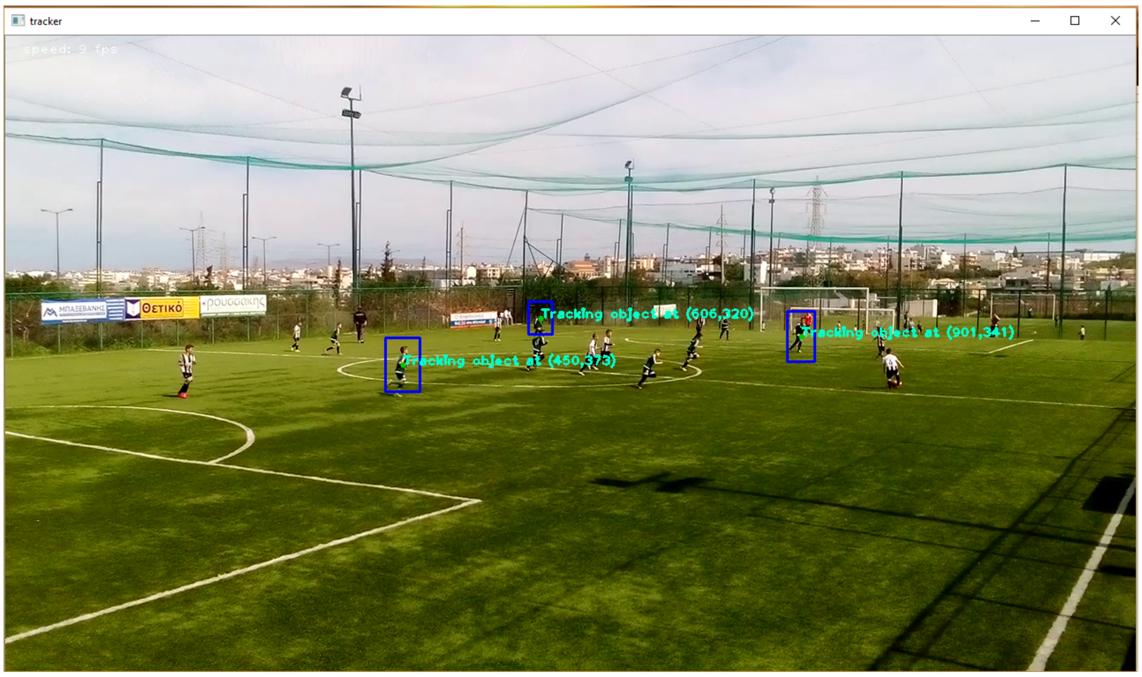

tracker
speed: 9 fps
Tracking object at (606,320)
Tracking object at (901,341)
Tracking object at (450,373)

Figure 4.3: Multiple football players tracking process

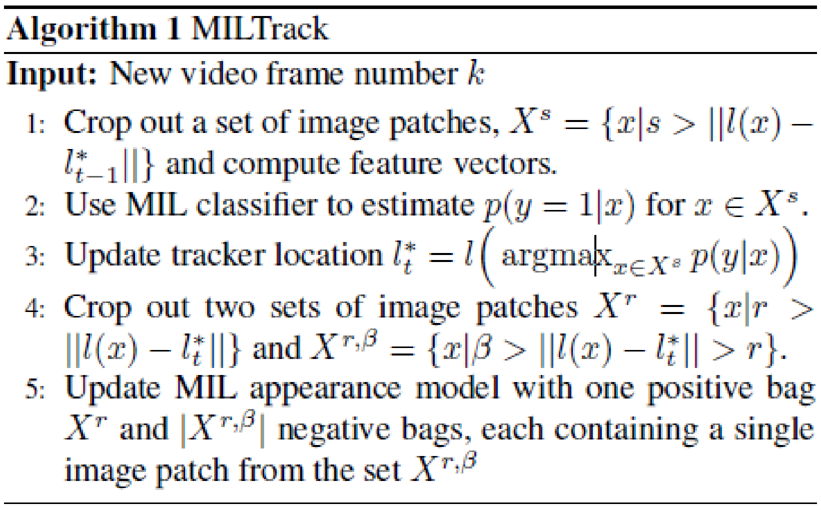
## 4.3 Analysis of selected algorithms

The goal of tracking is to find a moving object in the current frame, using the prior information of the motion parameters in the previous frame. Knowing the parameters of the motion model- the speed, the location and the position of the moving target- in previous frames, would enable to predict the new location. However, we do not only have the knowledge of the object's motion, we also know how the object looks in each of the previous frames. Specifically, we can encode what the object looks like by formulating an appearance model. Consecutively, to predict accurately the location

of the moving object, the appearance model can be used to go through a narrow region of interest in the location that the motion model predicted. Considering, the dramatics changes an object's appearance can have, a lot of modern trackers use the appearance model as a classifier, that is trained online.
Classifiers categorize a rectangular region of an image as either a background or object. An image patch is taken as input and the classifier returns a result between 0 and 1 to point out the probability that this image patch includes the object. The result is 0 when the image patch is the background and 1 when the classifier is unquestionably sure that the patch is the object. In machine learning, when 'online' is mentioned, algorithms are trained using a very few examples at run time, whilst a great number of examples train offline classifiers.
A classifier is trained by feeding its positive and negative examples, the object and the background. For instance, if we need to build a classifier for detecting football players, we train it with thousands of images containing football players and thousands of images that do not contain football players. This is the way that the classifier learns to discriminate what is a football player and what is not, although it is not feasible to have a great number of positive and negative classes, when we decide to build an online classifier. How the selected tracking algorithms approach this problem of online training, is investigated in the following section.

## 4.4 Multiple Instance Learning Tracker (MIL)

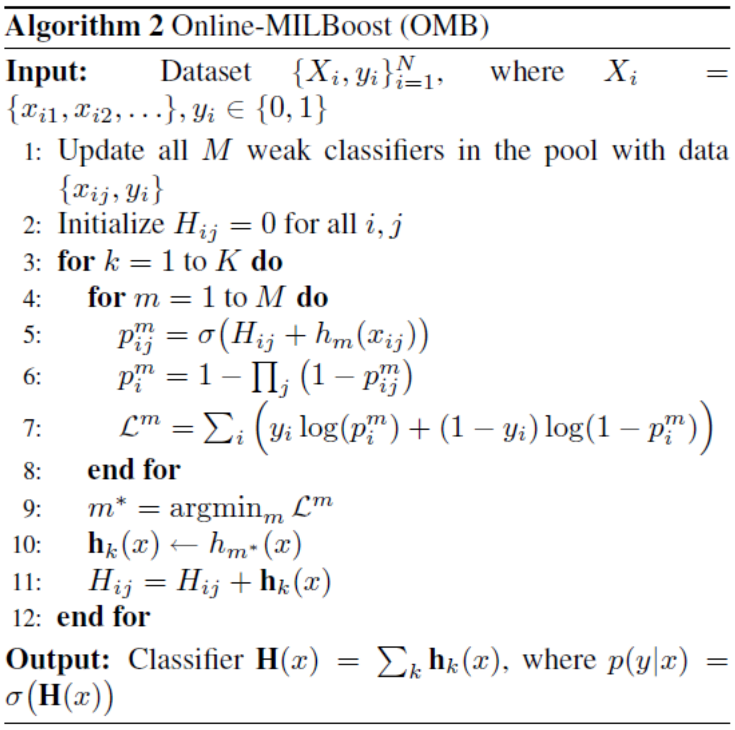
Tracking an object in a video, without knowing any other information, except its given location in the first frame, is a challenging problem, trying to be solved with tracking by detection techniques. These approaches separate the object from the background, by training online a discriminative classifier. The classifier extracts negative and positive examples from the current frame, using the current tracker state to bootstraps itself [23]. However, the classifier is drifting and degrading, whenever the training samples are wrongly labeled by the trackers unreliability. In this paper, authors presented MILTrack, a tracking system that make use of a novel online Multiple Instance Learning algorithm. The basic idea of their proposal is that they consider only the current location of the object, as a positive example. MIL tracking system generates several potential positive examples by searching in a small neighborhood around the current location. In MIL, the examples presented in sets called bags and labels are not provided for individual instances, only for bags. It is not necessary for all the images in the positive bag to be positive examples. They assume that if there is a labeled positive bag, it contains at least one positive instance, if not then the bag is negative. The centered patch on the objects current location and a small neighborhood around it, is included in a positive bag. They select MIL framework, which allow them to update the appearance model with a set of image patches, although the image patch that accurately captures the object they are interested in, is not known. As a result, they build an easy to implement and effective algorithm, with fewer tweaked parameters.



```
Algorithm 1 MILTrack
Input: New video frame number $k$
1: Crop out a set of image patches, $X^s = \{x | s > ||l(x) - l^*_{t-1}||\}$ and compute feature vectors.
2: Use MIL classifier to estimate $p(y = 1|x)$ for $x \in X^s$.
3: Update tracker location $l^*_t = l\left(\operatorname{argmax}_{x \in X^s} p(y|x)\right)$
4: Crop out two sets of image patches $X^r = \{x | r > ||l(x) - l^*_t||\}$ and $X^{r,\beta} = \{x | \beta > ||l(x) - l^*_t|| > r\}$.
5: Update MIL appearance model with one positive bag $X^r$ and $|X^{r,\beta}|$ negative bags, each containing a single image patch from the set $X^{r,\beta}$
```

Figure 4.4: Pseudocode of MIL tracker 1 [23]

Their tracking system includes the following components, image representation, appearance model and motion model. For the image representation, in every image patch, there is a vector of Haar-like features, which are randomly generated. Their system requires that each weak classifiers $hk$ is composed of a Haar-like feature $fk$ and parameters $\mu 1$, $\sigma 1$, $\mu 0$ can be updated online. Authors based their proposal on the boosting framework and the Online AdaBoost and its adaptation, to create a strong classifier, combining many weak classifiers.



```
Algorithm 2 Online-MILBoost (OMB)
Input: Dataset $\{X_i, y_i\}_{i=1}^{N}$, where $X_i = \{x_{i1}, x_{i2}, \ldots\}, y_i \in \{0, 1\}$
 1: Update all $M$ weak classifiers in the pool with data $\{x_{ij}, y_i\}$
 2: Initialize $H_{ij} = 0$ for all $i, j$
 3: for $k = 1$ to $K$ do
 4:   for $m = 1$ to $M$ do
 5:     $p_{ij}^{m} = \sigma\big(H_{ij} + h_m(x_{ij})\big)$
 6:     $p_i^m = 1 - \prod_j \big(1 - p_{ij}^m\big)$
 7:     $\mathcal{L}^m = \sum_i \Big(y_i \log(p_i^m) + (1 - y_i) \log(1 - p_i^m)\Big)$
 8:   end for
 9:   $m^* = \operatorname{argmin}_m \mathcal{L}^m$
10:   $\mathbf{h}_k(x) \leftarrow h_{m^*}(x)$
11:   $H_{ij} = H_{ij} + \mathbf{h}_k(x)$
12: end for
Output: Classifier $\mathbf{H}(x) = \sum_k \mathbf{h}_k(x)$, where $p(y|x) = \sigma\big(\mathbf{H}(x)\big)$
```

Figure 4.5: Pseudocode of Online MILBoost [23]

Each feature contains two to four rectangles, that have a real valued weight. This feature value is the pixels weighted sum in all the rectangles.

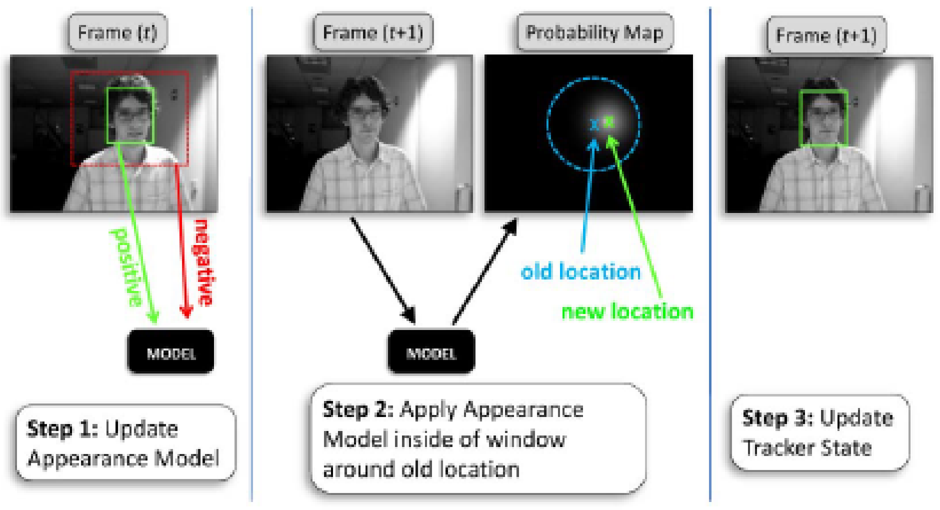

Frame (t)
positive
negative
MODEL
Step 1: Update Appearance Model
Frame (t+1)
Probability Map
old location
new location
MODEL
Step 2: Apply Appearance Model inside of window around old location
Frame (t+1)
Step 3: Update Tracker State

Figure 4.6: Tracking by detection with a greedy motion model [23]

A discriminative classifier is composing the appearance model, that estimates $p(y = 1|x)$, where x is the images patch representation in feature space. A binary variable y that belongs to $0, 1$ specifies the existence of the object we are interested in that image patch. In MIL, the data is formed as $\{(x1, y1), , (xn, yn)\}$ and the bag labels are defined as

$$y_i = max(y_{ij} \qquad (2)$$

The tracker maintains the object location lt, at every time step t for computational efficiency and simplicity reasons. A set of image patches, is cropped out for each new frame, as long as they are within some search radius s of the current location of the tracker. The radius s is for gathering positive instances during initialization. In order to update the trackers location a greedy strategy is used

$$l_t^* = l(\arg_{x \in X^s} maxP(y|x) \qquad (3)$$

to avoid maintaining the location of the target at every frame. They alternatively choose a motion model, where the trackers location at every step time t is equally expected to appear at trackers location $(t-1)$, within of course in a specified radius $s$.

$$P(l_t^*|l_{t-1}^*) \propto \begin{cases} 1 & \text{if } ||l_t^* - l_{t-1}^*|| < s \\ 0 & otherwise \end{cases} \tag{4}$$

The algorithm in MIL is trained with labeled bags. So, as soon as the set of patches $X_r$ is cropped out, this bag is positive labeled where r is smaller than radius $s$. From an annular region $X_r$,$b$, they cropped out patches to label negatives bag, where $b$ is another scalar and r is the same as before. A random subset of these image patches is labeled as negatives, due to the fact that this generates a large set.
Authors tested their MILTrack system in four publicly available grayscale video sequences resized to 320 × 240 pixels and four of their own. To demonstrate that MIL is stable and more robust tracker, they decided for all the experiments, that all algorithms parameters were fixed. They compare Online-Adaboost Tracker with positive radius $r = 1(OAB1)$ and $r = 5(OAB5)$, SemiBoost Tracker, FragTrack and MILTrack with radius 5. The scalar $\beta$ for sampling negative examples was set to 50 and the learning rate $\gamma$ for the weak classifiers was set to 0.85. The number of chosen weak classifier K was 50 and the number of weak classifiers M was set to 250. In the sequences that present challenging scale, lighting and pose changes, their proposed algorithm achieved the best performance. On the other hand, MILTrack even though, obtained better results than the compared trackers under the presence of partial occlusion, it could not handle full occlusions.

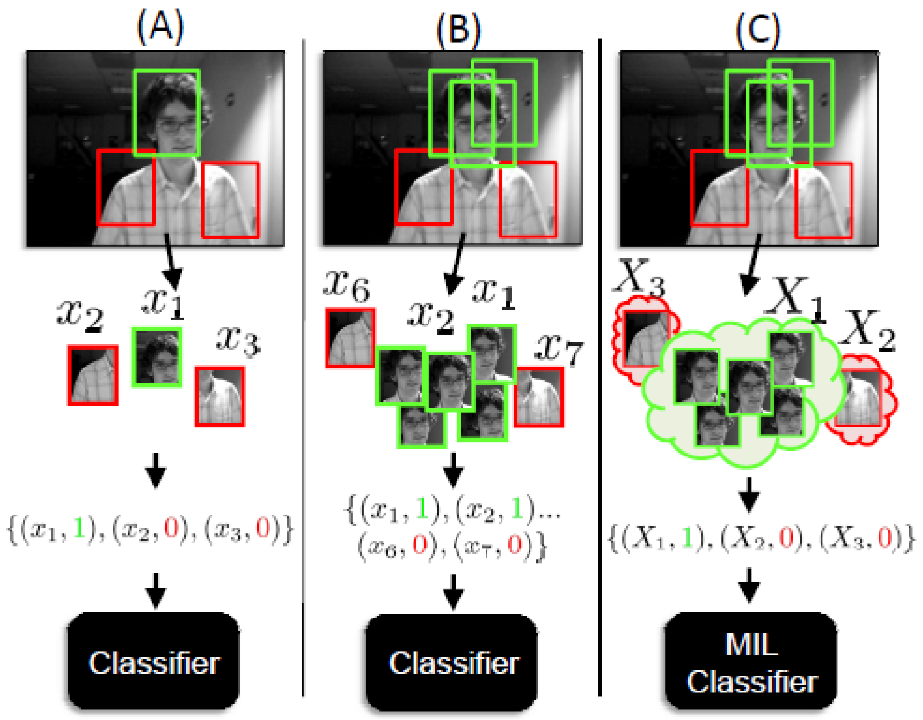

(A)
(B)
(C)
$x_2$
$x_1$
$x_3$
$x_6$
$x_2$
$x_1$
$x_7$
$X_3$
$X_1$
$X_2$
$\{(x_1,1),(x_2,0),(x_3,0)\}$
$\{(x_1,1),(x_2,1)\ldots(x_6,0),(x_7,0)\}$
$\{(X_1,1),(X_2,0),(X_3,0)\}$
Classifier
Classifier
MIL Classifier

Figure 4.7: Updating a discriminative appearance model
(A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can confuse the classifier causing poor performance. (C) Using one positive bag consisting of several image patches to update a MIL classifier [23]

## 4.5 Kernel Correlation Filters

A discriminative classifier that separates the surrounding environment from the target, is the basic element of most state of the art trackers. Natural image changes through time, so this classifier needs to be trained with sample patches, which are scaled and translated. Authors in this paper[24] proposed an analytic model for datasets of thousands of translated patches. In the Fourier domain, if a particular translations model is used, learning algorithms turn out to be easier as more samples are added. Correlation filters benefits from the fact that in the Fourier domain, the convolution of two patches is corresponding to an element-wise product. Thus, correlation filters can indicate linear classifiers output for image shifts or several translations straightaway. Authors focused on Ridge Regression with classical correlation filters and cyclically sifted samples, that enables fast learning with $O(n \log n)$ Fast Fourier Transforms. The objective of training is to find a function $f(z) = w^T z$ that minimizes the squared error over samples $x_i$ and their regression targets $y_i$

$$\min \sum_i (f(x_i) - y_i)^2 + \lambda ||W||^2 \tag{5}$$

To control overfitting, the regularization parameter $\lambda$ is used. The minimizer w is given by the following equation

$$W = (X^T X + \lambda I)^{-1} X^T y \tag{6}$$

where X is the data matrix, with one sample per row $x_i$ and a regression target $y_i$ is composed by each element of $y$. Quantities are for the most part complex valued, in the Fourier domain, so the above equation becomes

$$W = (X^H X + \lambda I)^{-1} X^H y \tag{7}$$

where $X^H$ is the Hermitian transpose.

The base sample is an $n \times 1$ vector representing the object of interest patch, denoted as x. The classifier needs to be trained with several virtual samples and the base sample, which must be translated.

The cyclic shift operator models one dimensional translations of this vector.

$$P = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{8}$$

To shift $x$ by one element, authors used product $P_x$, modeling a small translation.

$$P_x = [x_n, x_1, x_2, x_3, ..., x_{n+1}]^T \tag{9}$$

To achieve a larger translation, $u$ shifts are used by the matrix power $P^u x$. The same signal $x$ is obtained periodically every $n$ shifts, as result of the cyclic property. The full set of shifted signals is acquired with

$$\{P^u x | u = 0, ...., n-1\} \tag{10}$$

A regression with shifted samples, is then computed and the set of eq 5 is used to construct the rows of data matrix X

$$X = C(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & x_{n-3} \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{11}$$

All circulant matrices can be diagonalized using the Discrete Fourier Transform (DFT), aside from the generating vector x and this is expressed by the following equation

$$X = F(diag(\hat{x})F^H) \qquad (12)$$

$F$ is the constant matrix and does not depend on x, and $\hat{x}$ is the DFT of the generator factor $\hat{X} = F(x)$. To simplify the linear regression, when training data is being made of cyclic shifts, authors replace equation 7 with the no-centered covariance matrix $X^H X$,

$$X = F(diag(\hat{x^*})F^H F diag(\hat{x})F^H \qquad (13)$$

Vector $x$ is the first row in the below image.

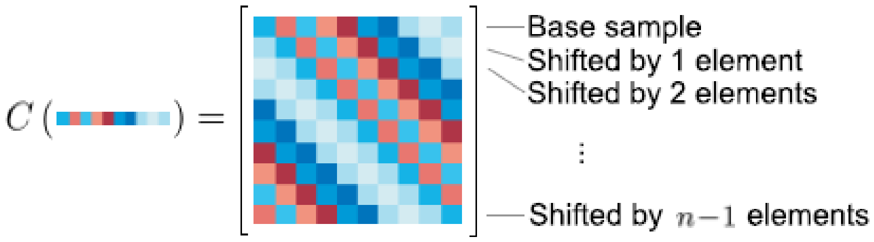


Figure 4.8: Illustration of circulant matrix- The rows are cyclic shifts of a vector image, or its translations in 1D. The same properties carry over to circulant matrices containing 2D images [24].

The Kernel trick allows more effective, nonlinear regression functions $f(z)$. The following steps map the inputs of a linear difficulty to an non linear feature space $\varphi(x)$.

- The solution w is expressed as linear combination of the samples

$$W = \sum_i \alpha_i \varphi(x_i) \qquad (14)$$

- The dot products are computed using the kernel function k

$$\varphi^T(x)\varphi(x') = k(x, x') \qquad (15)$$

- A $n \times n$ kernel matrix K is used to store the dot products between all pairs of the samples

$$K_{ij} = k(x_i, x_j) \qquad (16)$$

The dominance of the kernel trick is the fact that a vector does not need to be instantiated in the high dimensional feature space $\varphi(x)$. However, this causes a great number of samples. The following equation gives the kernelized version of Ridge regression solution.

$$\alpha = (K + \lambda I)^{-1} y \qquad (17)$$

Where a is the vector of coefficients $a_i$ and $K$ is the kernel matrix, that represent the solution in dual space. With the proof of the theorem that given circulant data $C(x)$, the corresponding kernel matrix $K$ is circulant if the kernel function satisfies $k(x, x') = k(Mx, Mx')$, for any permutation matrix $M$, they can diagonalize the above equation to achieve a fast result for the linear case obtaining

$$\hat{\alpha} = \frac{\hat{y}}{\hat{K^{xx}} + \lambda'} \qquad (18)$$

Where $k^{xx}$ is the first row of the kernel matrix $K = C(k^{xx})$, and a hat $\hat{}$ denotes the DFT vector. To detect the object of interest, the evaluation of regression function $f(z)$, needs to be done, for several candidates on several image locations. The kernel matrix between all candidates patches and all training samples is denoted as $K^z$. The patches are cyclic shifts of base patch z and base sample x, so each element of $K^z$ is given by $k(P^{i-1}z, P^{j-1}x)$ . The first row is needed to define the kernel matrix.

$$K^z = C(K^{xz}) \qquad (19)$$

Where $K^{xz}$ is the kernel correlation of $x$ and $z$. Afterwards, authors computed the regression function for all candidate patches with the following equation

$$f(z) = (K^z)^T \alpha \qquad (20)$$

And then, the diagonalization of $f(z)$ vector is occurred

$$\hat{f(z)} = \hat{K}^{xz} \odot \hat{\alpha} \qquad (21)$$

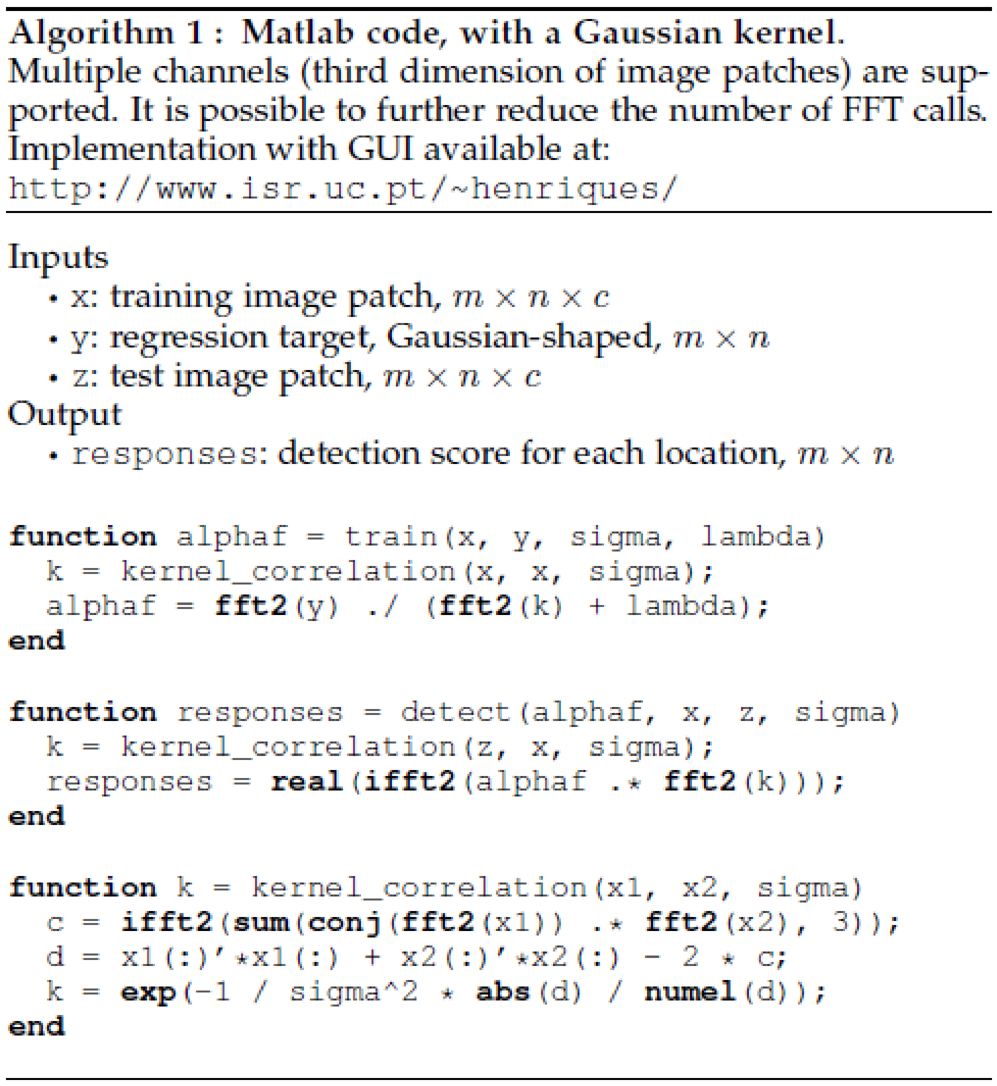
KCF is prepared to deal with multiple channels, as the 3rd dimension of the input arrays. It implements 3 functions: train (Eq. 18), detect (Eq. 21), and kernel_correlation (Eq. 22), which is used by the first two functions.

$$\hat{\alpha} = \frac{\hat{y}}{\hat{K^{xx}} + \lambda'} \qquad (18)$$

$$\hat{f(z)} = \hat{K}^{xz} \odot \hat{\alpha} \qquad (21)$$

$$K^{xx'} = exp\left(-\frac{1}{\sigma^2}\left(||x||^2 + ||x'||^2 - 2F^{-1}\left(\sum_c \hat{x_c^*} \odot \hat{x'}_c\right)\right)\right) \qquad (22)$$

The functionality of the KCF is presented in the figure below as Matlab code


**Algorithm 1 : Matlab code, with a Gaussian kernel.**
Multiple channels (third dimension of image patches) are supported. It is possible to further reduce the number of FFT calls. Implementation with GUI available at:
`http://www.isr.uc.pt/~henriques/`

Inputs
- `x`: training image patch, $m \times n \times c$
- `y`: regression target, Gaussian-shaped, $m \times n$
- `z`: test image patch, $m \times n \times c$

Output
- `responses`: detection score for each location, $m \times n$

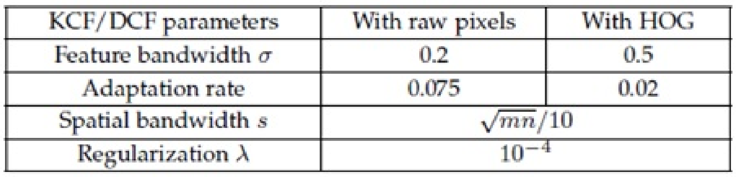
```
function alphaf = train(x, y, sigma, lambda)
  k = kernel_correlation(x, x, sigma);
  alphaf = fft2(y) ./ (fft2(k) + lambda);
end

function responses = detect(alphaf, x, z, sigma)
  k = kernel_correlation(z, x, sigma);
  responses = real(ifft2(alphaf .* fft2(k)));
end

function k = kernel_correlation(x1, x2, sigma)
  c = ifft2(sum(conj(fft2(x1)) .* fft2(x2), 3));
  d = x1(:)'*x1(:) + x2(:)'*x2(:) - 2 * c;
  k = exp(-1 / sigma^2 * abs(d) / numel(d));
end
```

Figure 4.9: Matlab code with a Gaussian kernel [24].

In the first frame, they train a model with the image patch at the initial position of the target. To make available some context, this image patch is larger than the target. The detection over the patch at the previous position is taking place, for each new frame. At that moment, targets position is updated, to the one that generated the maximum value. Finally, at the new position, a new model is trained and "linearly interpolate the obtained values of  and x with the ones from the previous frame, to provide the tracker with some memory" as authors pointed out. The proposed Kernelized Correlation Filter (KCF) approach, using a Gaussian kernel is implemented in Matlab. The algorithm is tested with two alternatives: one that takes the raw pixel values, and a second one that uses HOG descriptors with a cell size of 4 pixels, in particular Felzenszwalbs variant. Few parameters required by the KCF tracker, fixed for all videos and depicted in the following table.



| KCF/DCF parameters | With raw pixels | With HOG |
|---|---|---|
| Feature bandwidth $\sigma$ | 0.2 | 0.5 |
| Adaptation rate | 0.075 | 0.02 |
| Spatial bandwidth $s$ | $\sqrt{mn}/10$ | |
| Regularization $\lambda$ | $10^{-4}$ | |

Figure 4.10: Parameters used in all experiments. In this table, n and m refer to the width and height of the target, measured in pixels or HOG cells [24].

Authors compared their tracking approach with TLD and Struck tracker on fifty videos dataset instead of the original tests with 12 videos, using precision curve for the performance criteria. Moreover, instead of using raw pixels, they add to the KCF tracker a new variant founded on Histogram of Oriented Gradients (HOG) features. If the predicted target center is inside of ground truths distance threshold, then we consider that the frame is properly tracked. Precision curves demonstrate the percentage of correctly tracked frames for a range of distance thresholds.

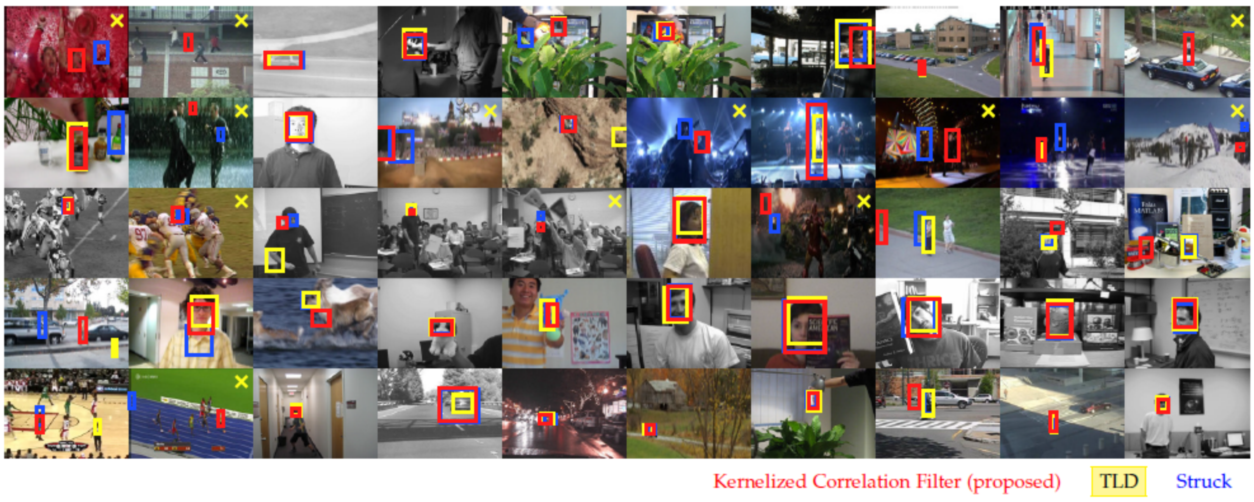


Figure 4.11: Qualitative results for the proposed Kernelized Correlation Filter (KCF)-compared with the top performing Struck and TLD. [24]

Without any feature extraction, the KCF accomplished better results than a linear filter [25]. With Histogram of Oriented Gradients features, nonlinear KCF surpassed TLD and Struck tracker. Tracker's speed is directly related to the size of the tracked region and is a significant factor when the comparison, between trackers based on correlation filter, is made. They decided to reduce their tracked region, to speed feature computation, but this decision slowed down the performance of the tracker.

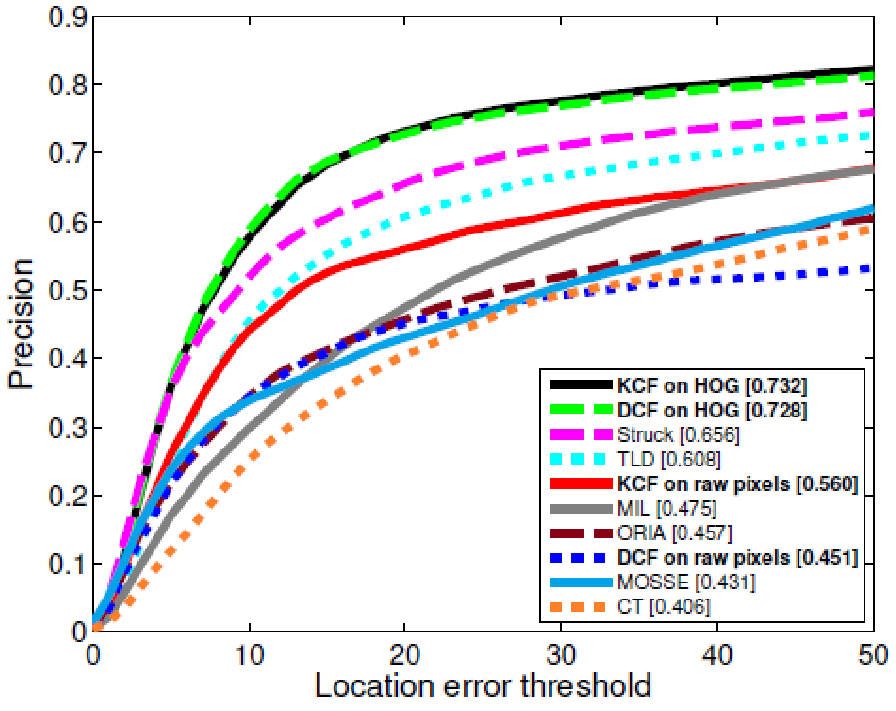


Figure 4.12: Precision plot for all 50 sequences [25]

Furthermore, they experimented with sequence attributes, such as occlusion and illumination changes, background clutter and out of view target. In occlusions and non-rigid deformations, KCF with HOG features surpassed the other algorithms. TLD's performance is better in out of view sequences, due to the fact that KCF lacks a failure recovery mechanism. Moreover, this algorithm performed better in background clutter, while almost all of trackers are severely affected, although it does not recover though from full occlusion.

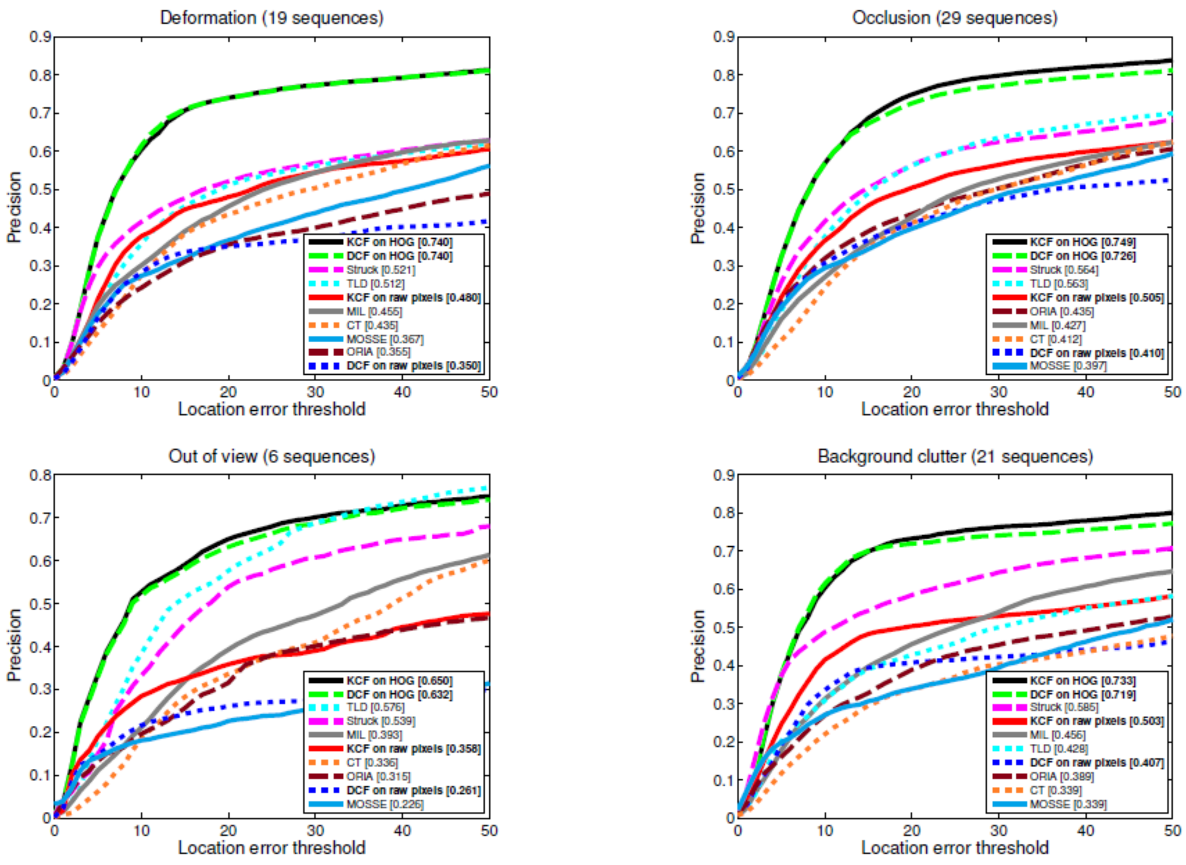


Figure 4.13: Precision plot for sequences with attributes [25]

In this paper, authors demonstrated that there is a possibility, multiple positive samples to have large overlapping regions. If specific conditions are met, kernel matrices and this overlapping data become circulant. DFT is applied to obtain their diagonalization to make the proposed tracker faster and more accurate at the same time. Their tracking framework was made open source, to encourage further developments.

# Chapter 5

# Algorithm evaluation

For our system, we choose a steady mobile camera, which only covers a part of the football field. The football field is mini $5 \times 5$ with dimensions $30 \times 60$ meters. Our application is implemented using OpenCV and C++ [26] . The input video sequences' resolution is 1280x720 size in pixels and the frame rate is 30 frames per second. The application is tested in 30 video sequences and the evaluation process is taking into consideration attributes- such as partial or full occlusion, out of view, where the players leave the camera field of view- which express the most common challenges of visual tracking. Kernel Correlation Filters and Multiple Instance Learning algorithms are selected. The video sequence is chosen and the application enables to define the bounding box that contains the preferable football player for the first frame. The tracker type is being initialized and the application goes through the video's frames, while the iteration process updates the tracker to capture for the current frame the new bounding box. Finally, the application displays the results.

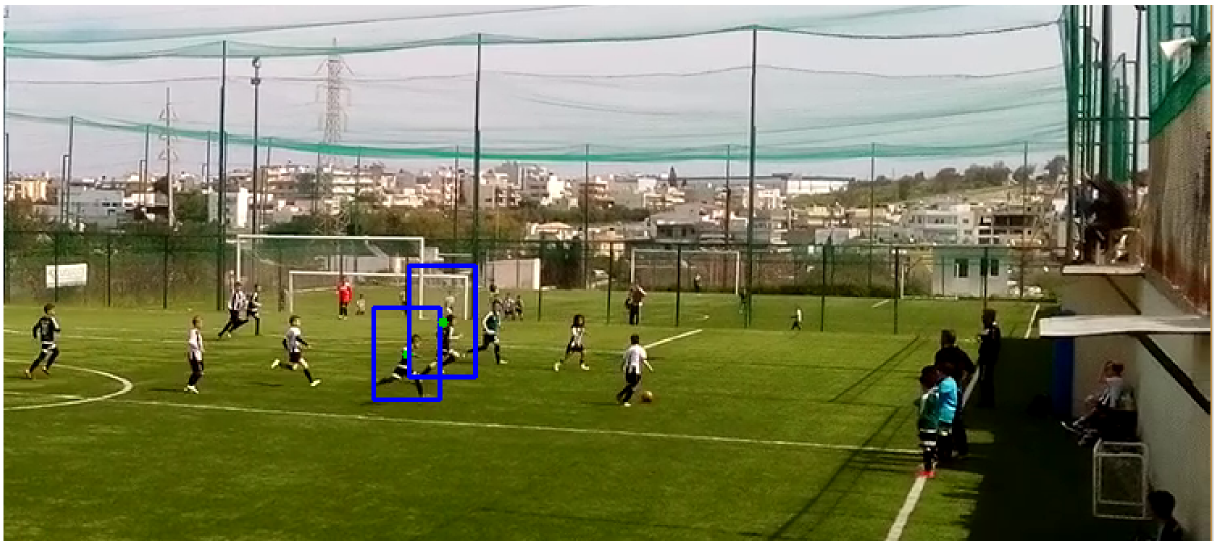


Figure 5.1: KCF under partial occlusion

KCF outperforms all the other tracking algorithms in speed and reliability, throughout the whole video sequence. Moreover, the selected algorithms were tested during partial and full occlusion. Two players are selected for the given sequence and in the following figures we can observe the results. KCF once more is faster in the multi players implementation and performs sufficiently under partial occlusion. As the authors of KCF tracker pointed, this algorithm cannot recover from full occlusion as we can observe in the figure below.

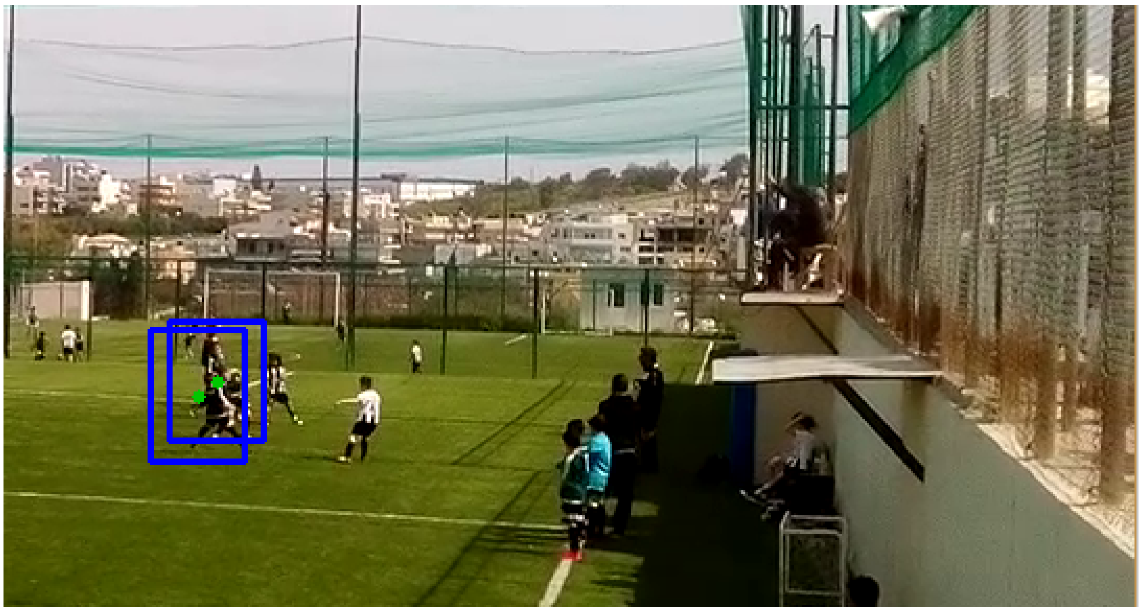


Figure 5.2: The two tracked players are fully occluded

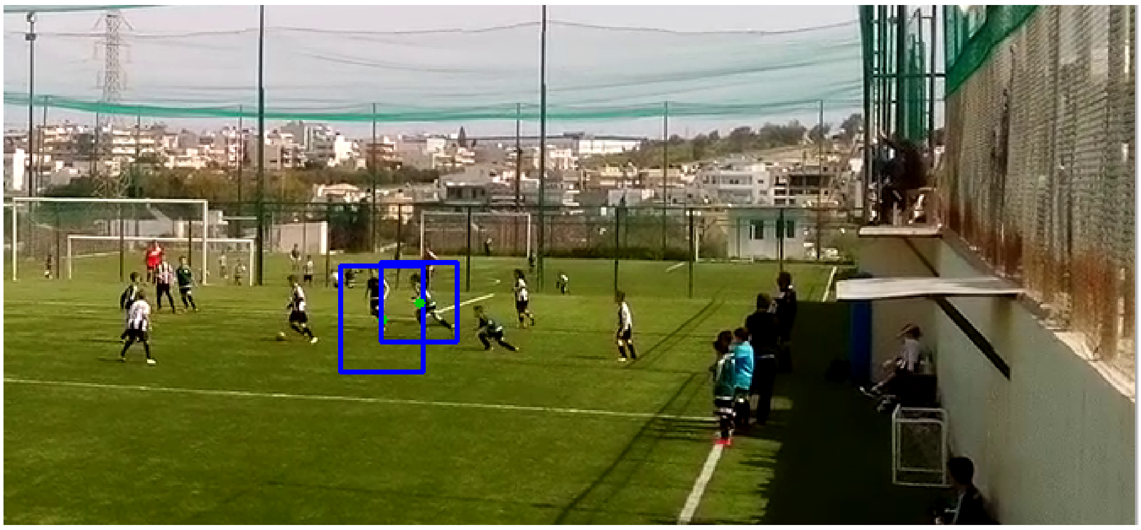


Figure 5.3: The KCF tracker does not recover from full occlusion

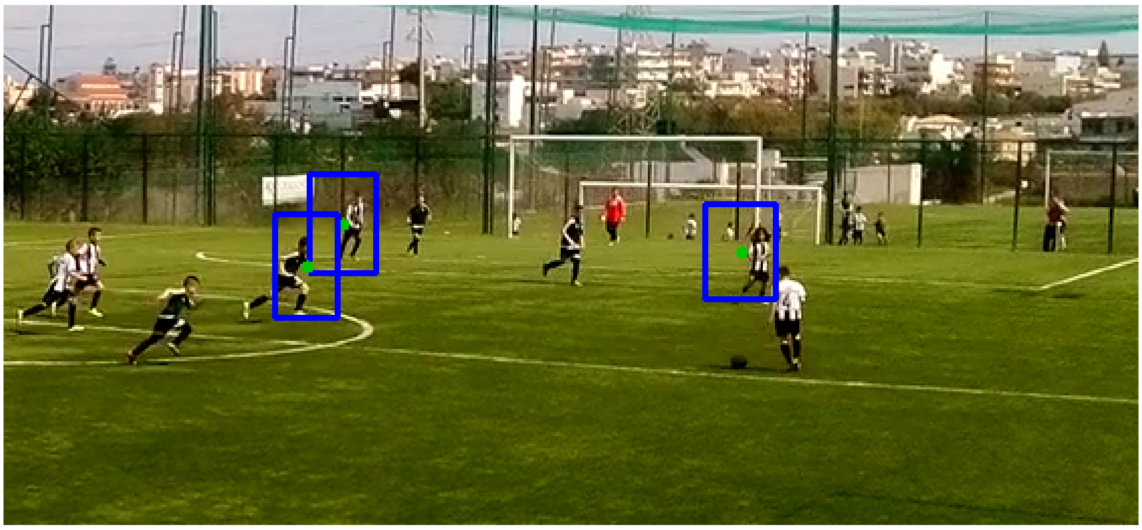


Figure 5.4: When the full occlusion is about to happen

MIL is reasonable efficient under partial occlusion and it keeps tracking the selected targets, however when the players are fully occluding one another, this tracking algorithm fails.

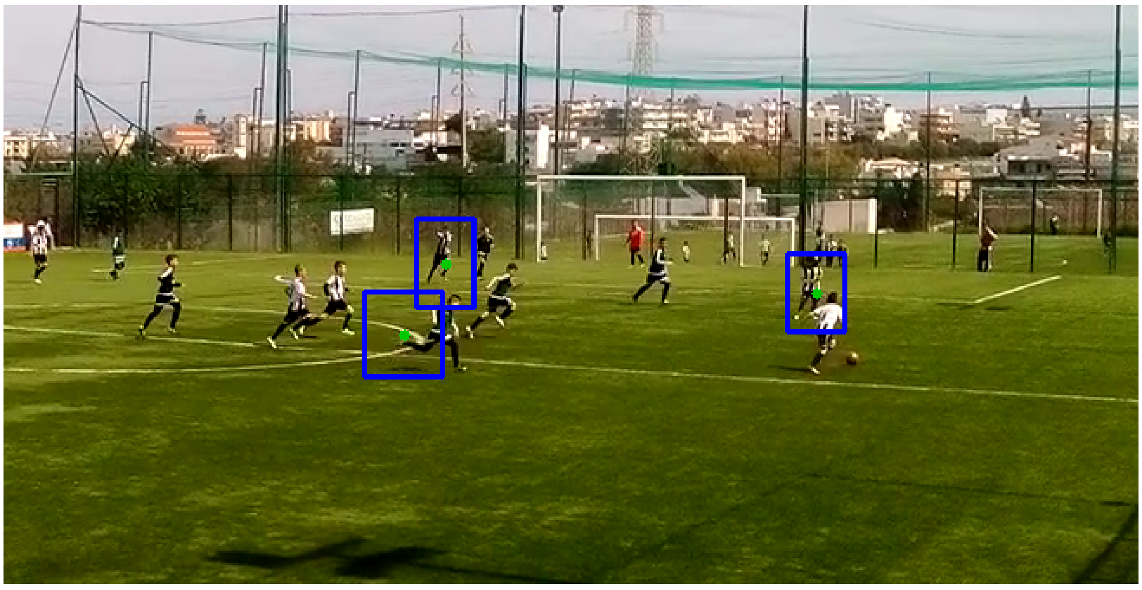
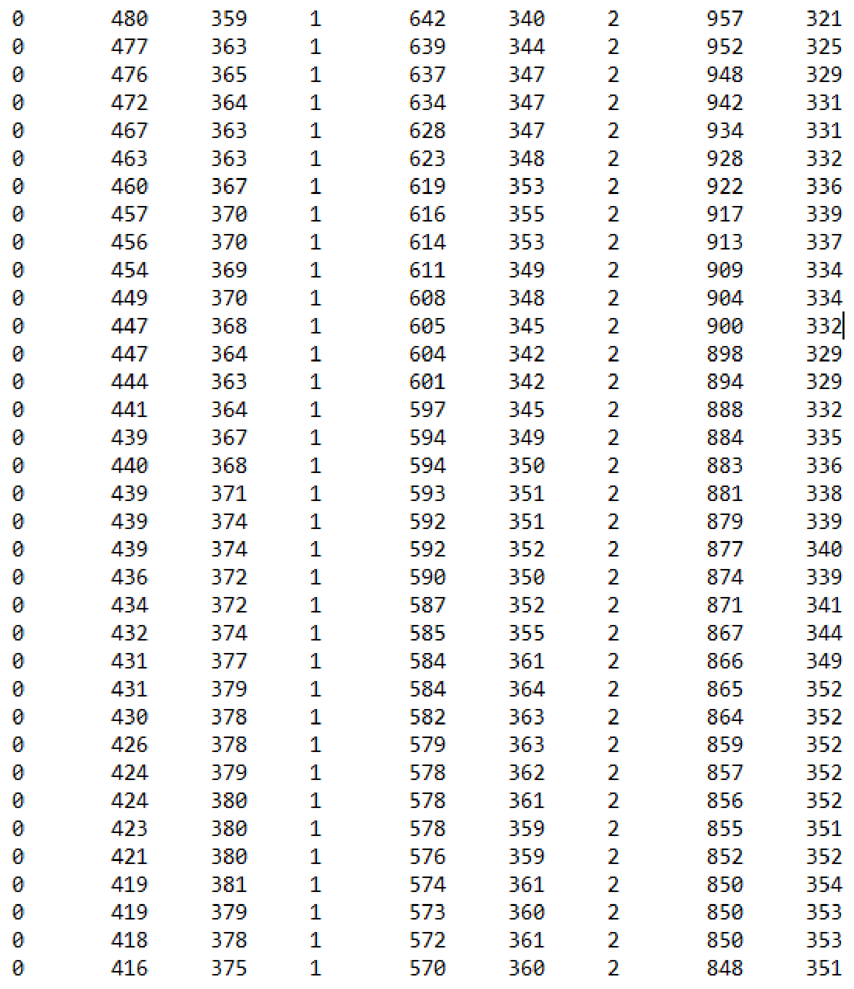


Figure 5.5: MIL fails under full occlusion

In the following figure, the output text file is illustrated, that stores the motion information of the three selected players to be tracked.



```
0       480     359     1       642     340     2       957     321
0       477     363     1       639     344     2       952     325
0       476     365     1       637     347     2       948     329
0       472     364     1       634     347     2       942     331
0       467     363     1       628     347     2       934     331
0       463     363     1       623     348     2       928     332
0       460     367     1       619     353     2       922     336
0       457     370     1       616     355     2       917     339
0       456     370     1       614     353     2       913     337
0       454     369     1       611     349     2       909     334
0       449     370     1       608     348     2       904     334
0       447     368     1       605     345     2       900     332
0       447     364     1       604     342     2       898     329
0       444     363     1       601     342     2       894     329
0       441     364     1       597     345     2       888     332
0       439     367     1       594     349     2       884     335
0       440     368     1       594     350     2       883     336
0       439     371     1       593     351     2       881     338
0       439     374     1       592     351     2       879     339
0       439     374     1       592     352     2       877     340
0       436     372     1       590     350     2       874     339
0       434     372     1       587     352     2       871     341
0       432     374     1       585     355     2       867     344
0       431     377     1       584     361     2       866     349
0       431     379     1       584     364     2       865     352
0       430     378     1       582     363     2       864     352
0       426     378     1       579     363     2       859     352
0       424     379     1       578     362     2       857     352
0       424     380     1       578     361     2       856     352
0       423     380     1       578     359     2       855     351
0       421     380     1       576     359     2       852     352
0       419     381     1       574     361     2       850     354
0       419     379     1       573     360     2       850     353
0       418     378     1       572     361     2       850     353
0       416     375     1       570     360     2       848     351
```

Figure 5.6: Output text file indicates the x and y position of the three selected players

The first column (0) is the first player, the second and third column is player's x and y position. The fourth column is the second player, the fifth and sixth column, is the 2nds player $x$ and $y$ position, the $7^{th}$ column is the third player, the $8^{th}$ and $9^{th}$ column is his $x$ and $y$ position.

## 5.1 Dataset

A thorough perfomance evaluation of tracking algorithms could be very complicated- as it is already mentioned in these papers [27] [28]- for the reason that tracking has many different applications in constantly changing environments. We can track people, faces and vehicles, while the presence of noise, scene illumination, object shape variation and total or partial occlusion are leading to significant problems that need to be overcomed. So, it is crucial to collect a very large dataset to evaluate and compare comprehensively tracking algorithms. Yi Wu et al.- in Visual Tracker Benchmark[30]- build a tracking dataset with 50 sequences with annotations. Using video sequences without labeled ground truth tends to make difficult the evaluation process, due to the fact that the reported result depends on object's location that are inconsistently annotated. When it comes to computer vision, ground truth data involves the set of images and its labels that define a model and includes the location, number and key features' relationship. The labeling of this set of images, can be added automatically (by image analysis) or manualy (by human) and then by using various machine learning methods, the model can be trained. Background, foreground or objects that are present in an image, form the labeled features of ground truth data. The following steps are utilized to create a ground truth dataset:

- Model design
- Training set
- Test set
- Classifier design
- Training and testing

"Unless the ground truth data contains carefully selected and prepared image content, the algorithms cannot be measured effectively. Better ground truth data will enable better analysis" [29].

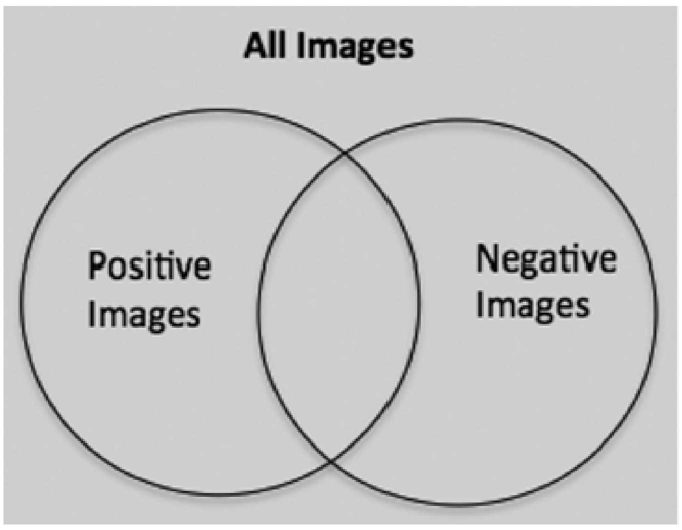


Figure 5.7: Set of all ground truth data, composed of both positive and negative training examples [29]

Considering the fact, that our results based on video sequences without labeled ground truth and different initial conditions and parameters, we decided to use the Visual Tracking Benchmark dataset with ground truth annotations [30] for comprehensive performance evaluation. The dataset contains 50 fully annotated- most commonly used- tracking sequences and the code library integrates the most publicly available tracking algortihms. Figure 5.8- 5.9, illustrates the first frame of each sequence where the target object is initialized with a bounding box. The sequences on the top left are more difficult for tracking than the ones on the bottom right of the figure. All sequence names are in CamelCase without any underscores or blanks. When there exist multiple targets each target is identified as dot+id number (e.g. Jogging.1 and Jogging.2). Each row in the ground-truth files represents the bounding box of the target in that frame, (x, y, box-width, box-height). In most sequences the first row corresponds to the first frame and the last row to the last frame, except the following sequences David(300:770), Football1(1:74), Freeman3(1:460), Freeman4(1:283).

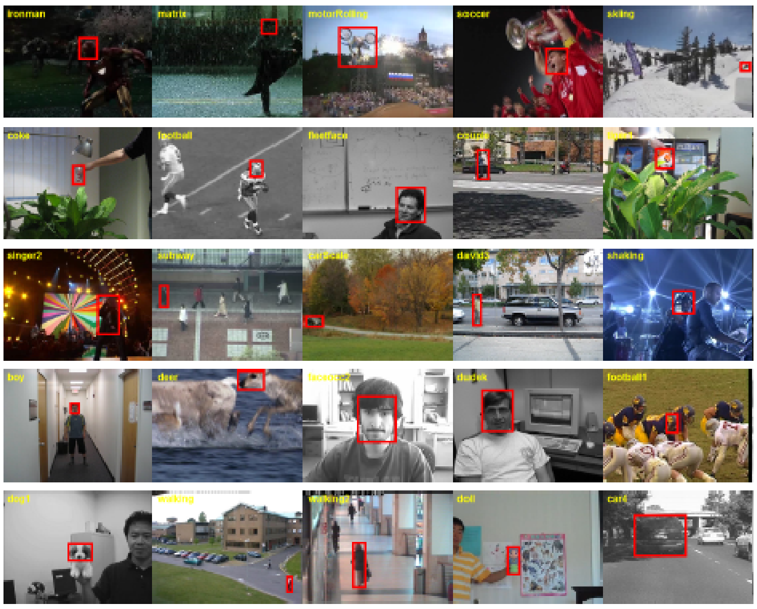


Figure 5.8: Tracking sequences for evaluation [30]

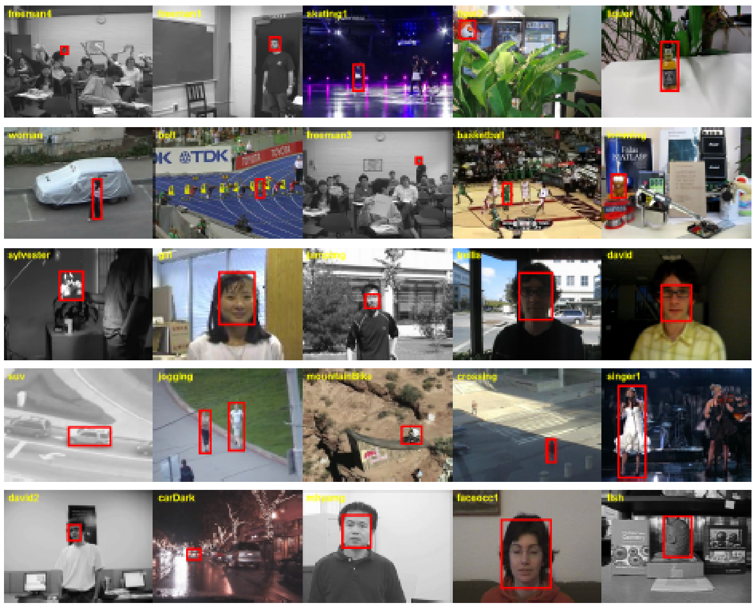
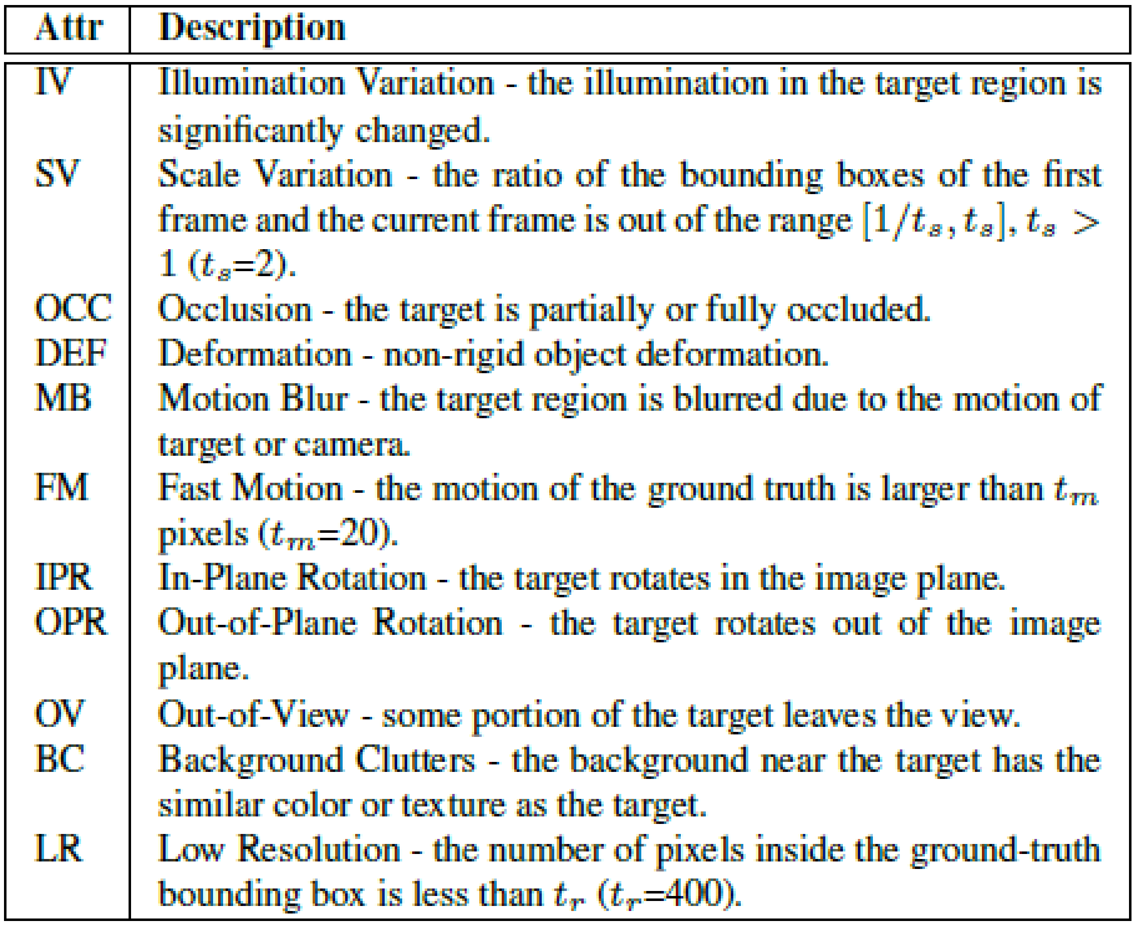


Figure 5.9: Tracking sequences for evaluation [30]

Each video in the dataset has been described by listing major challenges in computer vision tracking systems, such as illumination and scale variation, occlusion, fast motion etc.. Tracking performance can be affected by many factors, therefore to evaluate effectively the strength and weakness of tracking algorithms, Visual Tracking Benchmark annotates 11 attributes to the dataset sequences. In the following figure 5.10, some of the most common challenges in Visual Tracking are illustrated.



| Attr | Description |
|---|---|
| IV | Illumination Variation - the illumination in the target region is significantly changed. |
| SV | Scale Variation - the ratio of the bounding boxes of the first frame and the current frame is out of the range $[1/t_s, t_s]$, $t_s > 1$ ($t_s$=2). |
| OCC | Occlusion - the target is partially or fully occluded. |
| DEF | Deformation - non-rigid object deformation. |
| MB | Motion Blur - the target region is blurred due to the motion of target or camera. |
| FM | Fast Motion - the motion of the ground truth is larger than $t_m$ pixels ($t_m$=20). |
| IPR | In-Plane Rotation - the target rotates in the image plane. |
| OPR | Out-of-Plane Rotation - the target rotates out of the image plane. |
| OV | Out-of-View - some portion of the target leaves the view. |
| BC | Background Clutters - the background near the target has the similar color or texture as the target. |
| LR | Low Resolution - the number of pixels inside the ground-truth bounding box is less than $t_r$ ($t_r$=400). |

Figure 5.10: List of the attributes annotated to test sequences. The threshold values used in this work are also shown. [30]

## 5.2 Evaluation Methodology

Authors in [30], use the precision and success rate for quantitative analysis. Moreover, the evalutation of tracking algorithms' robustness, carried out in the following aspects.

- Success Plot
- Precision Plot

Success Plot (area overlap) refers to an evaluation metric, where the bounding box overlap is taking place. Given the tracked bounding box $r_t$ and the ground truth bounding box $r_a$, the overlap score is defined as

$$S = \frac{|r_t \bigcap r_a|}{|r_t \bigcup r_a|} \tag{23}$$

where $\bigcap$ and $\bigcup$ symbolize respectively the intersection and union of two regions, and $|.|$ is the pixels' number in the region. The perfomance measurement on frames' sequence, is calculated by counting the number of successful frames whose overlap S is larger than the given threshold $t_o$ (e.g. $t_o = 0.5$). The success plot shows the ratios of successful frames at the thresholds varied from 0 to 1. However, using a specific threshold, may not be representative for tracker evaluation. For showing tracking algorithms overall performance, authors in [27], instead use the Area Under Curve (AUC)- which is the success' rates average- of each success plot, rather than using one success rate value at a specific threshold.

Precision Plot refers to another widely used evalution metric, which is defined as the average Euclidean distance between tracked targets' center locations and the manually labeled ground truths and referred as center location error. The overall algorithm's performance is calculated by taking into consideration the average location error over all the frames of the selected sequence. For the precision plot, authors considerate a threshold value of 20 pixels. Precision plots are using only the bounding box locations and overlook its overlap and size, and that is the main reason that success plots are most preferable. The bounding box of the target (x, y, box width, box height) is represented by each row in the groundtruth files.

One- pass evaluation (OPE), the most common approach to evaluate tracking algorithms is- in the first frame- to initialize them from the ground truth position, run them through a test sequence and take into account the success rate or average position. Considering the fact, initialization may affects the performance of a tracking algorithm, Yi Wu et al.- in Visual Tracker Benchmark[30]- propose two methods to investigate tracker's robustness. They introduce Spatial Robustness Evaluation (SRE)- altering initialization spatially starting by different bounding boxes- and Temporal Robustness Evaluation (TRE)- altering initialization temporally starting at different frames. Figure 5.11 illustrates the different evaluation methods for trackers. Ground truth target location is represented by the green box and tracker's initialization is represented by the dotted boxes. The simplest method is OPE, where -in the first frame- we initialize the tracker and allow it to track the selected target throughout the whole sequence. With TRE method, the tracker is being initiated at different starting frames with the initialization of the ground truth bounding box. In SRE, the object states is reproduced by slightly shifting or scaling the ground-truth bounding box of a target object.

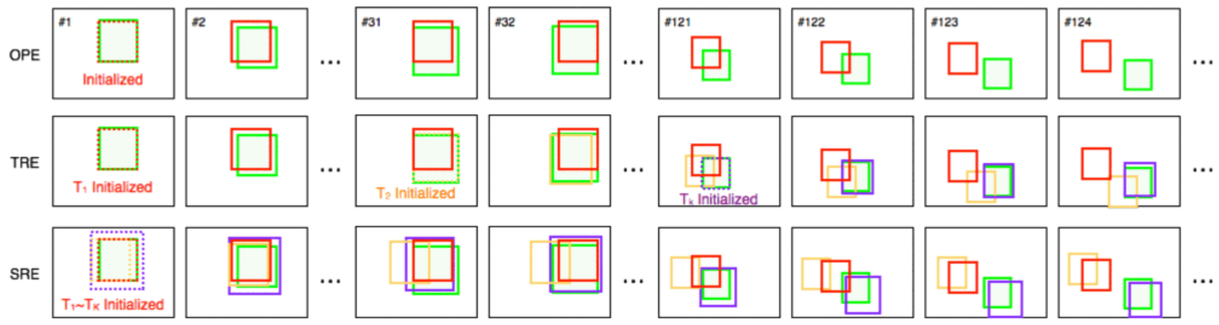


Figure 5.11: Evaluation methods of tracking algorithms

## 5.3 Evaluation Results

The source code default parameters are used in all evaluations for each tracker. For Spatial Robustness Evaluation, on each sequence we evaluate each tracker 12 times, where more than 350000 bounding box results are generated. For Temporal Robustness Evaluation, we partition each sequence in to 20 segments, so each tracker is perfomed on 310000 frames. The complete perfomance for all the trackers is encapsulated by the precision and success plots. For precision plots, we use the results at error threshold of 20 for ranking the trackers (e.g., this is the percent of frames for which the tracker was less than 20 pixels off from the ground truth). As the trackers tend to perform well in shorter sequences, the average of all the results in TRE tend to be higher. The tracking result is a sequence of bounding boxes at each frame. For the x-axis of the plot (overlap threshold)- the ratio of the tracked frames' bounding box has more overlap with the groundtruth bounding box than the threshold- and forms the success ratio. The values in the brakets in the figures are the AUC (area under curve), each of which is the average of all success rates at different thresholds when the thresholds are evenly distributed.

Figure 5.12, illustrates the overall performance of tracking algorithms. The red line describes KCF tracker with score 0.479, the pink dotted line is Boosting with score 0.422, the green dotted line is MIL with score 0.417, the black line illustrates MedianFlow tracker with score 0.359 and the blue dotted line depicts TLD tracker with score 0.292. In the success plots, the top ranked tracker KCF in TRE outperforms Boosting by 5.7 % , MIL by 6.2 %, MedianFlow by 12.1 % and TLD by 18.7 %.

In the precision plot, the performance score of KCF is 0.681, for MIL is 0.573, for Boosting 0.554, for MedianFlow 0.454 and for TLD 0.433. The top ranked tracker is once more KCF and outperforms MIL by 10.8 % , Boosting by 13.7 %, MedianFlow by 22.7 % and TLD by 24.8 %. MIL's score seems to be better, but this is due to the fact that the average error value does not measure the tracking performance correctly, because the output location can be random, when the tracking algorithm loses track of a target object. The overall performance - which is measured in the success plots- is more accurate than the measurement of the score at one threshold, thus the precision plots are used as supplementary.

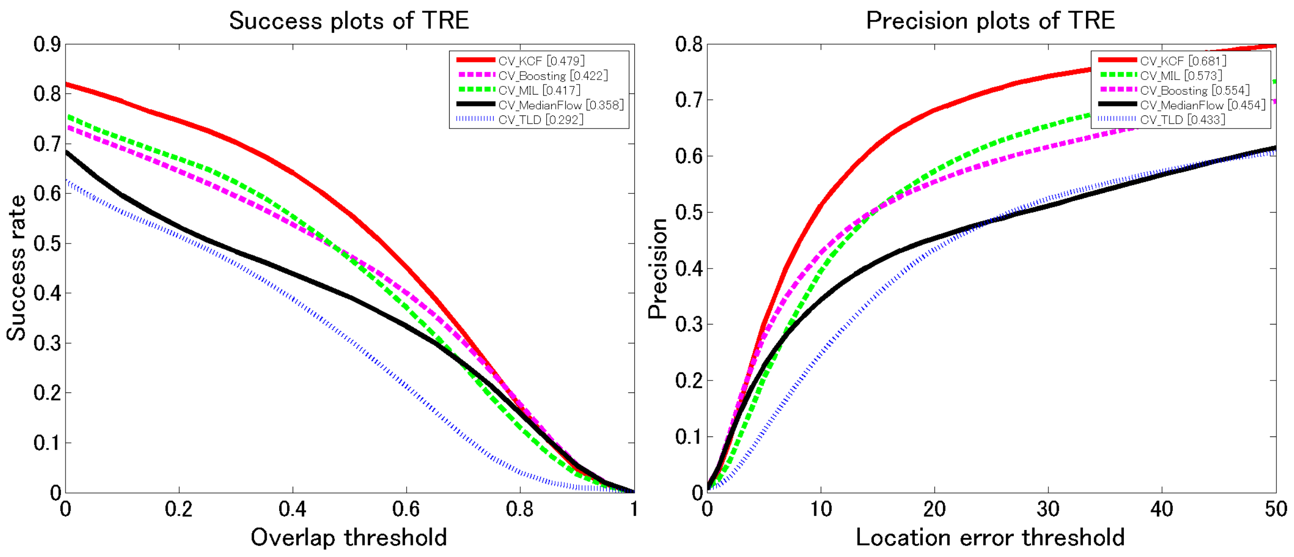


Figure 5.12: The Temporal Robustness Evaluation of the overall performance of tracking algorithms. The values in the legend are the performance scores. In the Success plots - the x axis (overlap threshold)- is the ratio of the frames whose tracked bounding box has more overlap with the ground truth bounding box than the threshold. The precision plot shows the ratio of successful frames whose tracker output is within the given threshold (x-axis of the plot, in pixels) from the ground-truth, measured by the center distance between bounding boxes.

Occlusion is one of the most challenging difficulty in computer vision systems. It is occured under three categories.

- Self occlusion
- Inter object occlusion
- Object to background occlusion

Self occlusion indicates that one part of an object is occluded be another part of the same object from a specific viewpoint, e.g. football player's feet. Inter object occlusion occurs as a result of the full or partial overlapping of more than one objects. We refer to object to background occlusion, when the tracked object is occluded by the background.

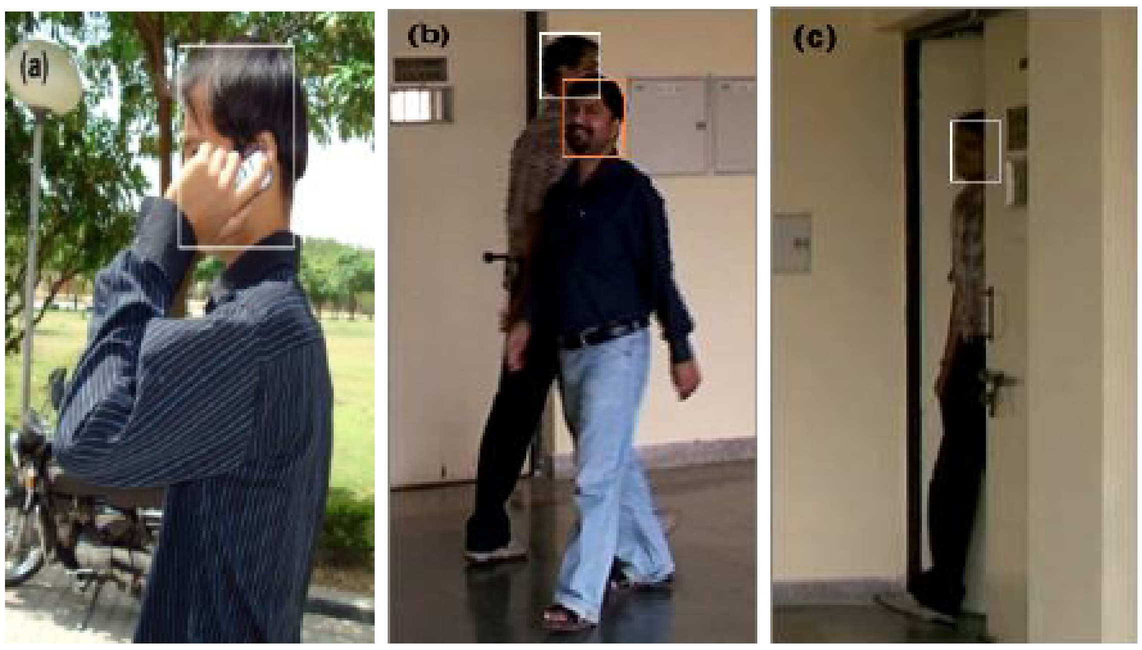


Figure 5.13: The three categories of occlusion types: a) Self Occlusion b) Inter object occlusion c) Object to Background occlusion [31]

Figure 5.14, illustrates the performance of tracking algorithms under the presence of occlusion. The red line describes KCF tracker with score 0.536, the pink dotted line is Boosting with score 0.438, the green dotted line is MIL with score 0.432, the black line illustrates MedianFlow tracker with score 0.339 and the blue dotted line depicts TLD tracker with score 0.282. Once more KCF outperforms Boosting by 9.8 % , MIL by 10.4 %, MedianFlow by 19.7 % and TLD by 25.4 %. In the sequences that present challenging scale, lighting and pose changes, MIL achieved the best performance. On the other hand, MIL even though, obtained good results than the compared trackers under the presence of partial occlusion, it could not handle full occlusions.

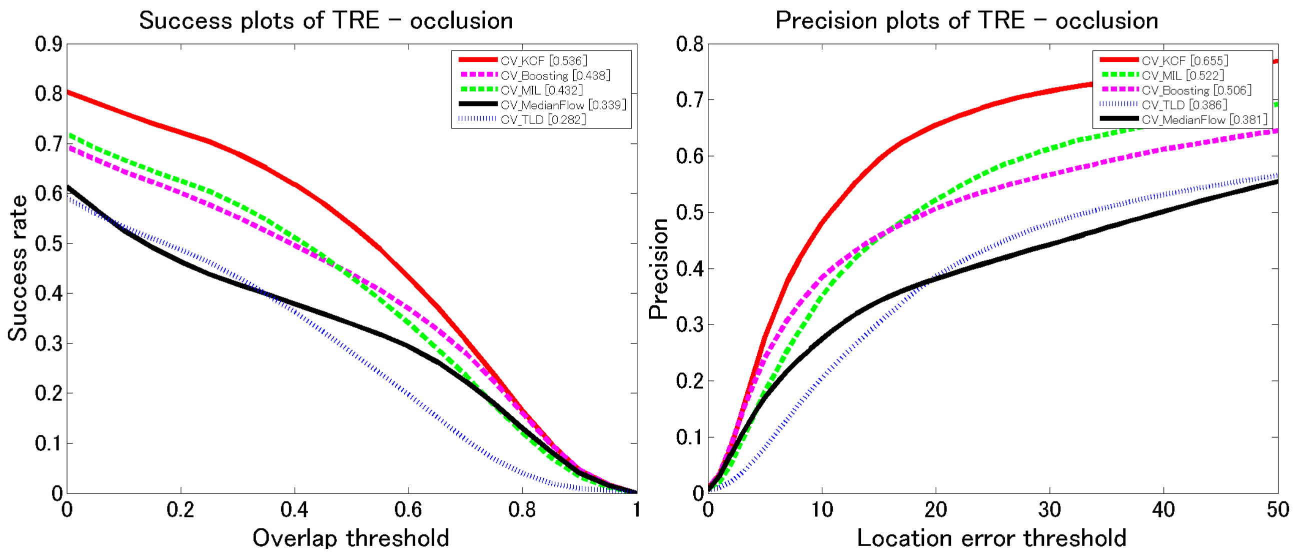


Figure 5.14: Success and Precision plot under the presence of occlusion

Tracker's speed is directly related to the size of the tracked region and is a significant factor when the comparison, between trackers based on correlation filter, is made. If we decide to reduce our tracked region, to speed feature computation, this will lead to slow down the performance of the tracker. The performance evaluation is running on a PC with Intel i5-3350 CPU @ 3.10 GHz with 16 GB of RAM.

## 5.4 Converting 2D motion into 3D world coordinates

The motion statistics and specifically the x and y position of the tracked players, is extracted in an output file. Our goal will be to exploit this information to represent the perspective video tracking coordinates to orthographic 3D coordinates in the 3D space. For our system, some known parameters are existing. The focal length of the camera is 29mm, the camera height in relation to the terrain is 5 meters, the resolution of our video is $1280 \times 720$ pixels and $y1$ is the center of the height of the tracked player. We transform $u, v$ coordinates of the camera to correspond to the world coordinates. Moreover, the position in 3D space is going to be displayed in respect to the upper left corner. The dimension of the objects is not mandatory, we only interested in object's position. To determine which screen $x$-coordinate corresponds to a point at $Ax$, $Az$, we need to multiply the point coordinates by:
$Bx = Ax\frac{Bz}{Az}$ (24),
where $Bx$ is the screen $x$ coordinate, $Ax$ is the model $x$ coordinate, $Az$ is the subject distance and Bz is the focal length.

In our case, the equation 24 becomes

$$U = f * \frac{x}{|z|} \qquad (25)$$
$$V = f * \frac{y}{|z|} \qquad (26)$$

for the $U, V$ coordinates respectively.

$y_1$ is 5000 milimeters (the camera's distance from the ground), and to calculate the unknown parameters of $x$ position and $z$ coordinate (depth) in 3D space, the equations becomes

$$x_1 = U_1 * \frac{|z_1|}{f} \qquad (27) \qquad \text{solution for finding x coordinate}$$
$$z_1 = y_1 * \frac{f}{v_1} \qquad (28) \qquad \text{solution for finding z1 coordinate}$$

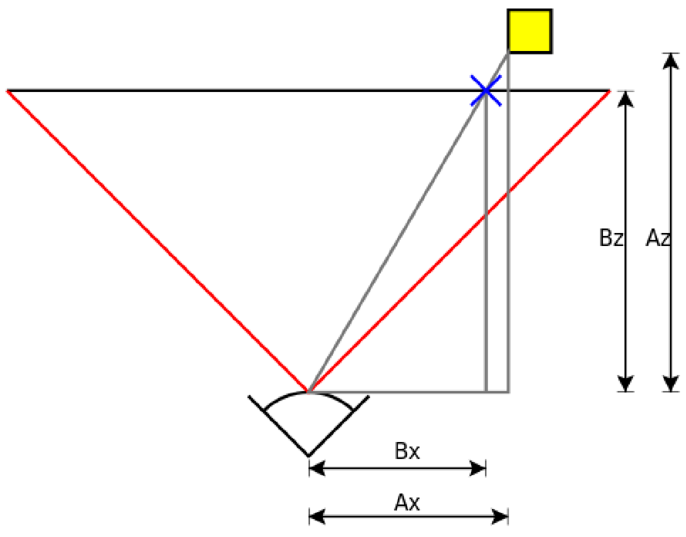


Figure 5.15: 3D projection diagram [34]

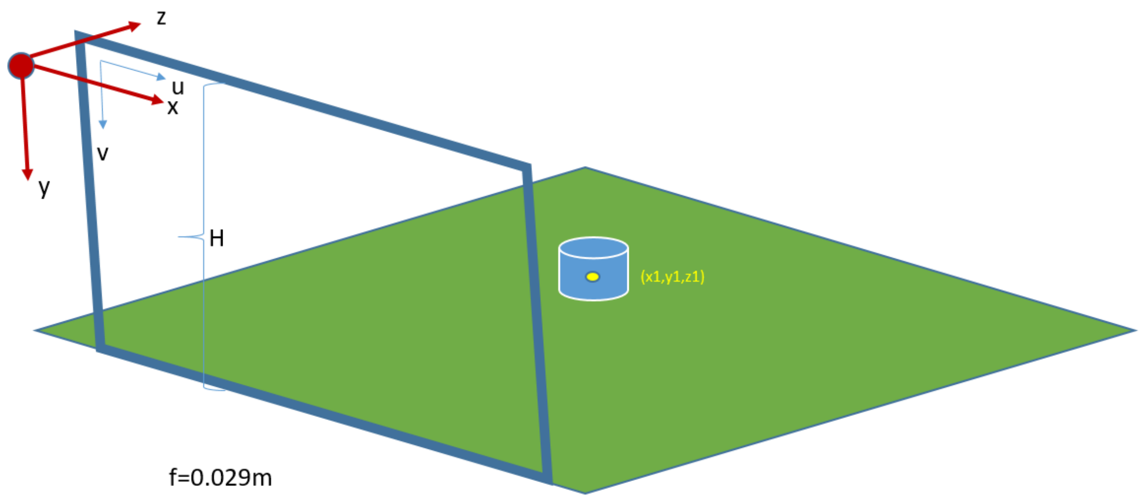


Figure 5.16: Transform the perspective video tracking coordinates to orthographic 3D coordinates

Figure 5.17 illustrates the position of three selected players in the first frame of our application. The application enables to define the bounding box that contains the preferable football player for the first frame. The tracker type is being initialized and the application goes through the video's frames, while the iteration process updates the tracker to capture for the current frame the new bounding box.

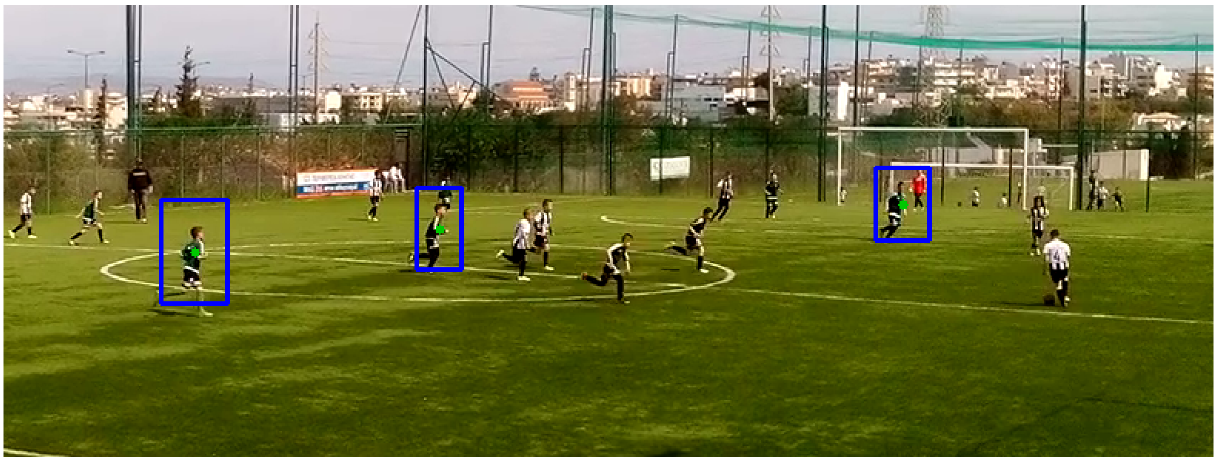


Figure 5.17: The position of 3 players in the first frame

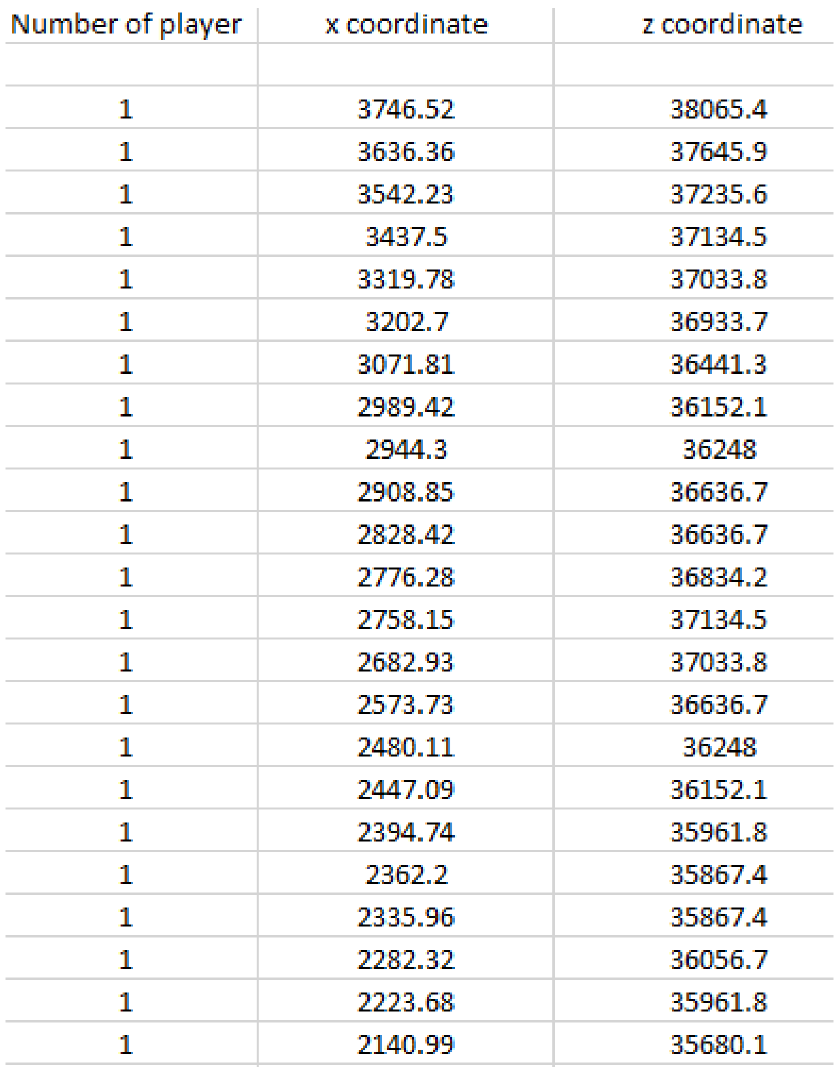
The motion results of each tracked player are calculated and the position of the players in the 3D world is stored in an output file.

The tracked object's distance is calculated, by using the Euclidean distance formula. In a 3 dimensional plane, the distance between points (X1, Y1, Z1) and (X2, Y2, Z2) is given by:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

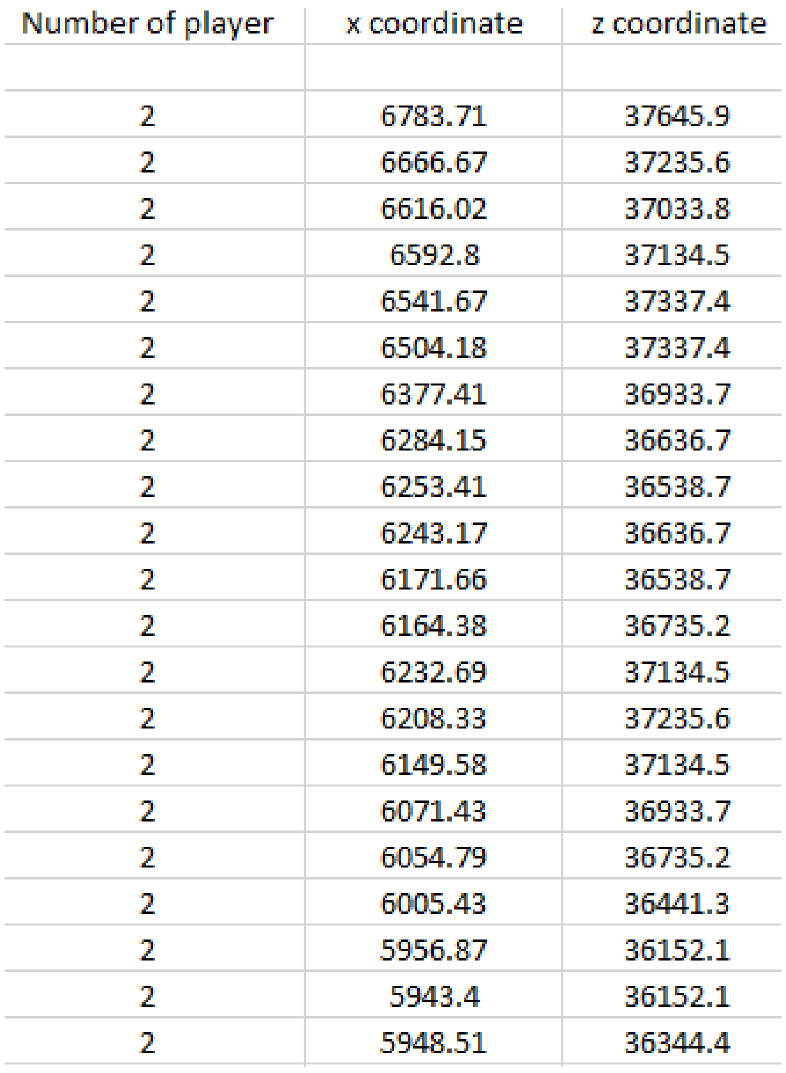
In our case XYZ values are in millimeters. To calculate the speed, we divide the distance traveled by the moving object with the time taken to travel distance 'd'. The average speed is the total distance traveled divided by the total time taken. For example, for some of our distance values, an average speed would be 4268 mm/sec.

In the following figures is illustrated the number of the first player, the x and z position in the 3D world (in milimeters).



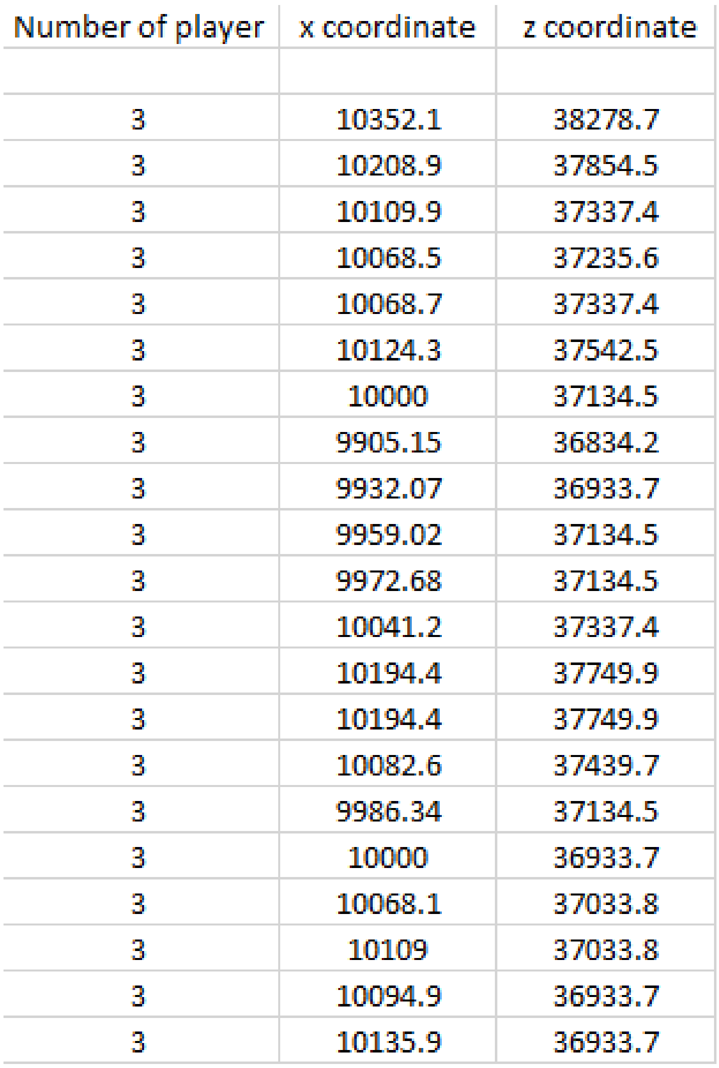
| Number of player | x coordinate | z coordinate |
|---|---|---|
| | | |
| 1 | 3746.52 | 38065.4 |
| 1 | 3636.36 | 37645.9 |
| 1 | 3542.23 | 37235.6 |
| 1 | 3437.5 | 37134.5 |
| 1 | 3319.78 | 37033.8 |
| 1 | 3202.7 | 36933.7 |
| 1 | 3071.81 | 36441.3 |
| 1 | 2989.42 | 36152.1 |
| 1 | 2944.3 | 36248 |
| 1 | 2908.85 | 36636.7 |
| 1 | 2828.42 | 36636.7 |
| 1 | 2776.28 | 36834.2 |
| 1 | 2758.15 | 37134.5 |
| 1 | 2682.93 | 37033.8 |
| 1 | 2573.73 | 36636.7 |
| 1 | 2480.11 | 36248 |
| 1 | 2447.09 | 36152.1 |
| 1 | 2394.74 | 35961.8 |
| 1 | 2362.2 | 35867.4 |
| 1 | 2335.96 | 35867.4 |
| 1 | 2282.32 | 36056.7 |
| 1 | 2223.68 | 35961.8 |
| 1 | 2140.99 | 35680.1 |

Figure 5.18: The position of the first player in the 3D world, for the first 23 frames



| Number of player | x coordinate | z coordinate |
|---|---|---|
| | | |
| 2 | 6783.71 | 37645.9 |
| 2 | 6666.67 | 37235.6 |
| 2 | 6616.02 | 37033.8 |
| 2 | 6592.8 | 37134.5 |
| 2 | 6541.67 | 37337.4 |
| 2 | 6504.18 | 37337.4 |
| 2 | 6377.41 | 36933.7 |
| 2 | 6284.15 | 36636.7 |
| 2 | 6253.41 | 36538.7 |
| 2 | 6243.17 | 36636.7 |
| 2 | 6171.66 | 36538.7 |
| 2 | 6164.38 | 36735.2 |
| 2 | 6232.69 | 37134.5 |
| 2 | 6208.33 | 37235.6 |
| 2 | 6149.58 | 37134.5 |
| 2 | 6071.43 | 36933.7 |
| 2 | 6054.79 | 36735.2 |
| 2 | 6005.43 | 36441.3 |
| 2 | 5956.87 | 36152.1 |
| 2 | 5943.4 | 36152.1 |
| 2 | 5948.51 | 36344.4 |

Figure 5.19: The position of the second player in the 3D world, for the first 23 frames



| Number of player | x coordinate | z coordinate |
|---|---|---|
| | | |
| 3 | 10352.1 | 38278.7 |
| 3 | 10208.9 | 37854.5 |
| 3 | 10109.9 | 37337.4 |
| 3 | 10068.5 | 37235.6 |
| 3 | 10068.7 | 37337.4 |
| 3 | 10124.3 | 37542.5 |
| 3 | 10000 | 37134.5 |
| 3 | 9905.15 | 36834.2 |
| 3 | 9932.07 | 36933.7 |
| 3 | 9959.02 | 37134.5 |
| 3 | 9972.68 | 37134.5 |
| 3 | 10041.2 | 37337.4 |
| 3 | 10194.4 | 37749.9 |
| 3 | 10194.4 | 37749.9 |
| 3 | 10082.6 | 37439.7 |
| 3 | 9986.34 | 37134.5 |
| 3 | 10000 | 36933.7 |
| 3 | 10068.1 | 37033.8 |
| 3 | 10109 | 37033.8 |
| 3 | 10094.9 | 36933.7 |
| 3 | 10135.9 | 36933.7 |

Figure 5.20: The position of the third player in the 3D world, for the first 23 frames

## 5.5 Conclusion

The purpose of this thesis was to examine if it is possible to develop an application for effective object tracking and detection in complex scenes, specifically in a football game. This thesis has shown that this is undoubtedly achievable with adequate precision and accuracy in the chosen video sequences. The implementation was accomplished with the C++ interface of open source library OpenCV 3.2 inside the Microsoft Visual Studio and the Multiple Instance Learning (MIL) and Kernel Correlation Filter (KCF) online training tracking algorithms were tested. he presence of noise, scene illumination, object shape variation and total or partial occlusion lead to significant problems in computer vision systems and make efficiently tracking process into a challenging task. Football player tracking and detection is a fundamental procedure in nearly all football video analysis. The results can be utilized at both a group and individual level. Football players can gain insight into their physiological performance, for instance player's covered distance extraction and trajectory. Coaches can extract information about the quality of tactical analysis and game strategy, strengths or weaknesses evaluation of the team's opponent or player and moreover, the verification of referee decisions. Furthermore, players' tracking information can be used by the broadcaster of a football match to provide enhanced replays and statistical analysis for the football's team supporter. The method can also be applied to a wide range of applications such as vision-based human-computer interaction. A huge amount of work is still required to be done, to show that this implementation is sufficient in a larger scale during a live football game. All in all this thesis has shown that the effectiveness of OpenCV with the combination of novel tracking algorithms has a huge potential and can circumvent the arising difficulties.

# Bibliography

[1] Zheng Pan, Shuai Liu1, Weina Fu A review of visual moving target tracking Accepted: 26 May 2016, Springer Science+Business Media New York 2016.

[2] ”human retina” Online. Available: https://www.eurekalert.org/pub_releases/2006-07/uops-prc072606.php Accessed March 2017.

[3] Hou Z, Han C (2006) A survey of visual tracking. Acta Automatica Sinica 32(4):603617.

[4] P. G. O Donoghue, M. Boyd, J. Lawlor, and E. W.Bleakley. Time-motion analysis of elite, semi-professional and amateur soccer competition. volume 41, pages 1-12. TEVIOT SCIENTIFIC, 2001.

[5] A Review on Video-Based Human Activity Recognition [Online]. Available: http://www.mdpi.com/2073-431X/2/2/88/htm Accessed May 2017

[6] [Online] https://www.youtube.com/watch?v=m-_t_aIlQwg Accessed May 2017.

[7] Zhaoquan Cai, Shiyi Hu, Yukai Shi, Qing Wang, Dongyu Zhang, Multiple human tracking based on distributed collaborative cameras Multimed Tools Appl (2017)

[8] [Ching-Han Chen, Chien-Chun Wang, Miao-Chun Yan, Robust tracking of multiple persons in real-time video, Multimedia Tools Appl (2016)

[9] Sun L, Liu G (2009) Field lines and players detection and recognition in soccer video. In: Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 1924 April 2009.

[10] Vandenbroucke N, Macaire L, Postaire J-G (2003) Color image segmentation by pixel classification in an adapted hybrid color space. Application to soccer image analysis. Computer Vision Image Understanding.

[11] [Xizhou Zhuang, Xina Cheng, Shuyi Huang, Masaaki Honda, Takeshi Ikenaga, Motion Vector and Players Features Based Particle Filter for Volleyball Players Tracking in 3D Space, Advances in Multimedia Information Processing - PCM 2015. Lecture Notes in Computer Science.

[12] Jia Liu, Xiaofeng Tong, Wenlong Li, Tao Wang, Yimin Zhang, Hongqi Wang, Bo Yang, Lifeng Sun, Shiqiang Yang Automatic Player Detection, Labeling and Tracking in Broadcast Soccer Video.

[13] Choi K, Seo Y (2011) Automatic initialization for 3D soccer player tracking. Pattern Recogn .

[14] DOrazio T, Leo M, Spagnolo P, Mazzeo PL,Mosca N, Nitti M, Distante A (2009) An investigation into the feasibility of real-time soccer offside detection from a multiple camera system.

[15] M. Manafifard, H. Ebadi, H. Abrishami Moghaddam, Multi-player detection in soccer broadcast videos using a blob-guided particle swarm optimization method.

[16] Online,Available:https://www.computer.org/csdl/trans/tp/2013/07/ttp2013071704-abs.html[Accessed May 2017]

[17] Online,Available:http://grassroots.fifa.com/en/for-coach-educators/technical-elements-for-grassroots-education/laws-of-the-game-for-small-sided-formats/dimensions-of-the-pitch.html[Accessed May 2017]

[18] Introduction OpenCV Online Available: http://docs.opencv.org/3.2.0/d1/dfb/intro.html Accessed February 2017

[19] OpenCV contributes [Online]. Available: https://github.com/opencv/opencv_contrib [Accessed April 2017]

[20] OpenCV tracking [Online]. Available: http://docs.opencv.org/3.0-beta/modules/tracking/doc/tracking.html Accessed March 2017

[21] Long-term optical tracking UML [Online]. Available: http://docs.opencv.org/3.0-beta/modules/tracking/doc/tracking.html [Accessed May 2017]

[22] Long-term optical tracking UML [Online]. Available: http://docs.opencv.org/3.0-beta/modules/tracking/doc/tracking.html [Accessed May 2017]

[23] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In Computer Vision and Pattern Recognition, 2009. CVPR 2009.

[24] Joo F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, High-Speed Tracking with Kernelized Correlation Filters, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.

[25] KCF results [Online]. Available: http://www.robots.ox.ac.uk/ joao/circulant/ [Accessed May 2017]

[26] OpenCV download[Online]. Available: http://opencv.org/ [Accessed February 2017][ Microsoft Visual Studio [Online]. Available: https://www.visualstudio.com [Accessed February 2017]]

[27] Wu, Yi et al., "Object Tracking Benchmark", IEEE Trans. Pattern Anal. Mach. Intell. 37 (2015): 1834-1848.

[28] Wu, Ye et al., "Online Object Tracking: A Benchmark", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2411-2418.

[29] Ground Truth Data, Content,Metrics, and Analysis, Chapter 7

[30] "Visual Tracker Benchmark", Retrieved September 23, 2017, from http://cvlab.hanyang.ac.kr/tracker-benchmark/datasets.html

[31] Jalal A S, Singh J. The state-of-the-art in visual object tracking. Informatica Slovenia, 2012, 36(3): 227248

[32] B. Mao, J. Cao and Z. Wu, "Web-based Visualisation of the Generalised 3D City Models Using HTML5 and X3DOM," vol. 2, no. 5, pp. 349-359, 2012

[33] Jung, J. Behr, and H. Graf, X3DOM AS CARRIER OF THE VIRTUAL HERITAGE Fraunhofer Institut fr Graphische Datenverarbeitung, Darmstadt, Germany

[34] 3D Projection diagram [Online]. Available: https://en.wikipedia.org/wiki/3D_projection [Accessed May 2017]