UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

MULTIPLE MULTIPARTITE GRAPHS OCCLUSION SOLVER FOR SOCCER PLAYERS' TRACKING

Tesis sometida a la consideración de la Comisión del Programa de Estudios de Posgrado en Ingeniería Eléctrica para optar al grado y título de Maestría Académica en Ingeniería Eléctrica

LENNON NÚÑEZ MEOÑO

Ciudad Universitaria Rodrigo Facio, Costa Rica

2024

Dedication

To my parents and family, for always supporting me into achieving every goal.

To my managers, coworkers, and friends that helped me survive this adventure.

To God, for granting me challenges for growth.

Thanks

Acknowledgement

This research was supported by CeNAT - CONARE Fellowship Program, in collaboration with CNCA (National Advanced Computing collaboratory)

This research was partially supported by a machine allocation on Kabré supercomputer at the Costa Rica National High Technology Center

This thesis is approved by the Electrical Engineering Postgraduate Studies Committee of the University of Costa Rica, as a partial requirement to opt for the Master's degree in Electrical Engineering.

Esta tesis fue aceptada por la Comisión del Programa de Estudios de Posgrado en Ingeniería Eléctrica de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Académica en Ingeniería Eléctrica.

JAIME CASCANTE VINDAS (FIRMA)
PERSONA FISICA, CPF-01-1008-0549.
Fecha declarada: 24/03/2025 06:17:32 p. m.
Firmado digitalmente

Dr. Jaime Cascante Vindas
Dean Representative
Sistema de Estudios de Posgrado

Dr. rer. nat. Francisco Siles Canales
Thesis Director

ESTEBAN MENESES ROJAS (FIRMA)
Digitally signed by ESTEBAN MENESES ROJAS (FIRMA)
Date: 2025.03.19 14:15:31 -06'00'

Ph.D. Esteban Meneses Rojas
Thesis Assessor

MARVIN COTO JIMENEZ (FIRMA)
PERSONA FISICA, CPF-01-1065-0799.
Fecha declarada: 19/03/2025 02:07:06 p. m.
Esta es una representación gráfica únicamente, verifique la validez de la firma.

Ph.D. Marvin Coto Jiménez
Thesis Assessor

M. Sc. Marco Villalta Fallas
Director Representative
Programa de Posgrado en Ingeniería Eléctrica

Firmado por:
LENNON FERNANDO NUÑEZ MEOÑO (FIRMA)
Razón:
Fecha: 3/24/2025 9:33 PM

Lennon Núñez Meoño
Candidate

Contents

Abstract

Multiple Multipartite Graphs tracking, a derived approach by mixing Multiple Hypothesis Tracking with Multipartite Graphs, was integrated into ACE soccer player tracking system to address occlusion events. Additionally, a spurious filtering based on hue histograms and predominant color swatches evaluation was integrated to provide the tracking algorithms with a data set clean of spurious objects, which considerably and negatively affect the tracker performance. Combining this spurious filter with other methods to handle occlusion events, as occlusion alarm and merge blob splitting, it was possible to achieve a reduction of 88% in false positives and an increase of 25% in MOTA in the test data set. When evaluating the tracker in reduced sets to study occlusion events more closely, a MOTA of 76% was achieved, representing an improvement of 50% when compared against the original tracker with a MOTA of 50.55% under the same data set.

Resumen

Rastreo por Múltiples Grafos Multipartitos, derivado de mezclar rastreo por múltiples hipótesis con grafos multipartitos, fue integrado en la plataforma de rastreo de jugadores de fútbol ACE para enfretar eventos de oclusión. Adicionalmente, un filtro de espurias basado en histogramas del tono y la evaluación de colores predominantes en el muestreo de color fue integrado para proveer a los algoritmos de rastreo de conjuntos de datos libres de espurias, las que considerablemente afectan de manera negativa al rendimiento del rastreador. Al combinar este filtro de espurias con otros métodos para manejar eventos de oclusión, como lo son la alarma de oclusión y el separador de blobs fusionados, fue posible obtener una reducción del 88 % en los falsos positivos y un incremento del 25 % en MOTA. Al evaluar el rastreador en un conjunto de datos reducido para estudiar más de cerca los eventos de oclusión, se alcanzó un MOTA de 76 %, representando una mejora del 50 % al comparar con el MOTA de 50.55 % obtenido por el rastreador original para el mismo conjunto de datos.

List of Tables

List of Figures

Glossary

Bad-Behaved Scene Frame from a soccer match video recording in which player blobs are agglomerated in such a way it becomes .. xi, 113, 114, 117, 120, 121

Dynamic Occlusion Event Occlusion event in which the occlusion's Agglomeration Size is variable and/or the event is comprised of smaller intermittent occlusion events, e.g. new players interacting with previously occluded ones at any point of the event, players constantly entering and leaving the occlusion event area, two players occluding intermittently, etc. . 23, 69

False-new-player Case in which a player's identity is lost during a certain amount of frames, after which the same player is identified as a new one, therefore being assigned a new player identity label. E.g. have players identified as p1 and p2 during frames 40 to 45, during frames 46 to 50 p2 is not detected, but it is detected again as a player during frames 51 to 60. Due to being missing in frames 46 to 50, it is detected as a new player entering the scene, therefore, it is assigned label p3 instead of p2, with no connection in tracking history of labels p2 and p3.. 22

Identity Swap Case in which at least two players have their identity labels swapped. Usually occurs with players of the same team in which visual features and proximity lead the tracking algorithm to detect one as the other.. 7–9, 11, 12, 20, 22, 24

Medium-Well-Behaved Scene Frame from a soccer match video in which player blobs are somewhat spaced, tracking algorithm may or may not solve occlusion events.. 113, 117

Multiple Object Tracking Accuracy A performance metric used in computer vision and tracking systems to evaluate how accurately and consistently objects are tracked over time. It measures the overall quality of a tracking system by considering factors such as missed objects, false positives, and identity switches in the tracking process.. 11, 69

Multiple Object Tracking Precision A performance metric used to evaluate the precision of a multiple object tracking system. It measures the accuracy of the predicted object locations by calculating the average distance between the predicted and ground truth object positions.. 84

Occlusion Visual effect in which, due to the projection of a 3D scene into a 2D plane, the object of interest is visually blocked by another object from the same scene. Angle perspective of the camera and objects proximity affect occlusion perception. The occlusion is identified by its length (video frames) and event complexity.. 6–13, 15–17, 19–24, 112, 113

Occlusion Agglomeration Size Number of players involved in an occlusion event.. 23

Occlusion Dynamics Index Value that represents the complexity of the occlusion event's dynamics. It is calculated as the sum of the weights of different occlusion dynamics: intermittency, team interaction.. 23

Occlusion Event Complexity Score that depicts how complex an occlusion event is in terms of the involved objects' interaction and the occlusion area objects density. The event can be categorized by its Dynamics, as Steady Occlusion Event or Dynamic Occlusion Event, and by its Agglomeration Size, represented by the number of players involved. It's the product of the Occlusion Dynamics Index and its Agglomeration Size.. 23, 113

Occlusion Event Region Region in the image plane where the occlusion event occurs. Its location changes accordingly to the players location, and its size may vary if it is a Dynamic Occlusion Event.. 23

Steady Occlusion Event Occlusion event in which the occlusion's Agglomeration Size is constant, i.e. the same occluded players are present from start to end of the event, and there is no intermittency of players. . 23, 69

Well-Behaved Scene Frame from a soccer match video in which player blobs are spaced and may be present only simple occlusion events that the tracking algorithm can solve without needing an occlusion solver.. xii, 113, 117, 120, 122

Acronyms

AI Artificial Intelligence. 14

BPG Bipartite Graph. 10

CNCA National Advanced Computing Collaboratory (acronym in Spanish). 70

DOE Dynamic Occlusion Event. 23, 69, *Glossary:* Dynamic Occlusion Event

HOMHT Hypothesis Oriented Multiple Hypothesis Tracking. 54

IoU Intersection over Union. 40

KDE Kernel Density Estimation. 70, 122–124

MHT Multiple Hypothesis Tracking. ix, 6, 11–14, 31, 32, 34, 47, 48, 52, 54, 57, 62, 68, 72, 81, 84, 85, 88, 90, 112

MMPG Multiple Multipartite Graphs. ix, 6, 13, 14, 17, 32, 48, 55, 57, 62, 72, 80–82, 84, 86, 88–90, 124

MOT Multiple Object Tracking. 6, 8, 10, 17, 20, 62

MOTA Multiple Object Tracking Accuracy. 11, 69, 84–88, *Glossary:* Multiple Object Tracking Accuracy

MOTP Multiple Object Tracking Precision. 84, *Glossary:* Multiple Object Tracking Precision

MPG Multipartite Graph. 6, 10, 13, 14, 27–32, 34, 49–51, 54–57, 65, 90, 122, 123

OAS Occlusion Agglomeration Size. 23, 24, *Glossary:* Occlusion Agglomeration Size

ODI Occlusion Dynamics Index. 23, 24, *Glossary:* Occlusion Dynamics Index

ODI-I Occlusion Dynamics Index Intermittency weight. 23, 24

ODI-T Occlusion Dynamics Index Team weight. 24

OEC Occlusion Event Complexity. 23, 24, 113, *Glossary:* Occlusion Event Complexity

OEC Occlusion Event Region. 23, *Glossary:* Occlusion Event Region

OECS Occlusion Event Complexity Score. 23, 24, 69, 85

PDL Players Dispersion Level. 116, 117, 120, 123, 124

PF Particle Filter. 8, 9, 11, 12, 22, 23, 27, 38

PRIS-Lab Pattern Recognition and Intelligent Systems Laboratory. 18, 69, 70, 72, 74, 78

ROI Region of Interest. 12, 22

SOE Steady Occlusion Event. 23, 69, *Glossary:* Steady Occlusion Event

TOMHT Track Oriented Multiple Hypothesis Tracking. 54–56, 124

WSL Windows Subsystem for Linux. 48, 70

Nomenclature

AgIx	Agglomeration Index. It has a value of 1 if there are only two players present in the occlusion event, and the number of players involved in any other case.
CS	Complexity Score, calculated as $CS = DyIx * AgIx$
DyIx	Dynamics Index. The sum of Intermittency and team interaction weights. $DyIx = ODI_{team} + ODI_{inttermitency}$
G_i	Single graph, with position i inside a multipartite graph.
k	Frame number.
k_f	Final frame.
k_i	Initial frame.
mpg_{k_i,k_f}	Multipartite graph starting on frame k_i and ending on frame k_f.
N	Number of nodes.
n	Node of a graph.
O	Occlusion event.
o	Occlusion event snippet.
$O^{complexity}_{k_i,k_f}$	Occlusion event from frame k_i to k_f, and complexity score *complexity*.
$o_{k,objects}$	Occlusion event snippet in the frame k, involving *objects* number of individuals.
w	Window of frames used to perform object tracking in sections of the video.
w_K	Window of size k.
w_{k_i,k_f}	Window composed of the frames from k_i to k_f.

Examples:

- Let $mpg_{300,800}$ a multipartite graph of size 501 graphs, result of the tracking process from frames $k_i = 300$ to $k_f = 800$ in the video window $w_{300,800}$.
- The graph G_4, of the multipartite graph $mpg_{300,800}$, has $N = 22$ nodes, while the graph G_5 has $N = 18$ nodes, an error in the tracking is possible as less objects are tracked in the subsequent graph.

Chapter 1

Introduction

This chapter covers the project's background and motivation, problem description and proposal of the solution to the problem stated.

1.1 Background and Motivation

Object tracking is used in different areas where it is important to know the motion of the objects of interest; some examples are submarine tracking, where a submarine needs to recognize projectiles and other ships moving alongside using passive sonars, aircraft surveillance with a similar scenario but using passive radars or an electronic warfare device to recognize the moving surroundings [77], in analysis of sports data [92], and, more recently, in medicine fields for cell tracking [54].

In terms of sport disciplines, the athletes usually use visual guides to improve and practice, be it recordings of the sport in execution or other visual aids on details of the discipline. The traditional method to analyze all the information available from those sources is to manually pick the scenes of interest and then select key plays to replay and study, the coach then will design a strategy based on the information extracted to improve the athlete's performance. Object tracking is introduced as a way to automatize data extraction for sports analysis, some disciplines require the tracking of an athlete's motion, in other cases like team sports, it is required to capture the motion of the many players involved. Players tracking goal is to provide information of the position of the player at any time during the whole match, by analyzing this simple piece of data it is possible to tell different statistics of its performance: total time played, distance traveled, area of the field covered during the match, speed, fatigue and placement of the player during different key plays [69]. Thus, players' performance and tactical evaluation can be assessed based on information from object tracking, making it a valuable tool for sports analysis.

One of the most popular sports in the world, with an estimate of more than 3.5 billion of fans, is the Association Football, or simply Football or Soccer [97]. It is a game in which two teams of 11

players, using any part of their bodies except their hands and arms, try to maneuver the ball into the opposing team's goal area, with a valid maneuvering field of about 90–120 m long and 45–90 m wide. The team that scores more goals wins at the end of a match [72].

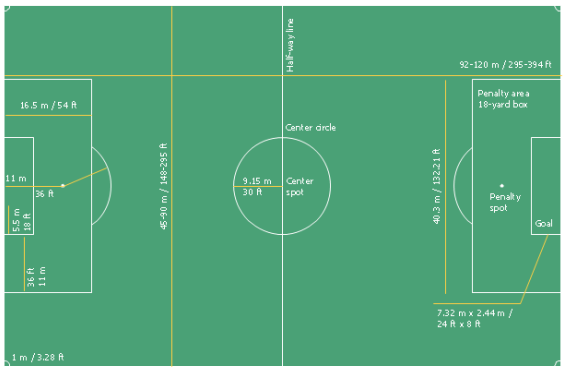


Figure 1.1: Football Soccer (Association Football) field. Source `https://www.conceptdraw.com/examples/football-pitch-dimensions-in-metres`

In the last years, some companies have dabbled into soccer players tracking for sports data analysis, the following are some examples of the most cited commercial applications in professional literature:

Kizanaro: It's a Uruguayan company started in 2008 as part of an academic project, with the purpose to provide services into soccer applied technology. It is used by the soccer teams of Angola, Uruguay, Paraguay, Venezuela, Colombia and others. The company states that it allows a fast visualization and analysis of the matches, obtaining information about the performance of the teams and players into video sections. The software classifies and breaks down the main plays of the match [41].

TRACAB: Optical Player Tracking solution from ChyronHego team, started in 2003 in Sweden. It's installed in about 300 soccer stadiums, being the official tracking technology used in leagues such as the English Premier League, German Bundesliga and Spanish La Liga, having also been selected for the major international UEFA and FIFA tournaments, as well as other sports than soccer. It uses camera arrays to record the field and delivers 3D information of the matches. The object tracking is supervised by a human operator (figure 1.2) [91].

Figure 1.2: TRACAB Soccer Players Tracking System: Camera array and supervised real time tracking. Source: https://chyronhego.com/products/sports-tracking/tracab-optical-tracking/

ChyronHego also offers others solution to sports analysis as an annotation tool, named Paint, used in sports news broadcasting to better visualize the different key plays. Nonetheless, the tool is a graphical visualization aid for sports journalists rather than a tracking or "ground truth" generator framework [63].

STATS: Company with more than 35 years in the market working with sport fans and athletes. The company offers different solutions into sports analysis and fan engagement, for data capture the system SportVU is used, which offers performance statistics via extraction and processing of players and ball coordinates during the match, using High Definition (HD) cameras and statistical software and algorithms [89], [90].

Match Analysis: Started as a way to share the coach's view to the team players, introduced into the market in 2000 in USA with data collection centrals in USA and Mexico. It brings a set of tools for video analysis and digital libraries to provide performance information and data storage, to perform tactical analysis of soccer teams. Real time data is collected and then synced with the video recordings by the analysts [51].

Since in 2015 the IFAB (International Football Association Board) allowed the use of techno-wearables during some soccer matches [23] as well as the standardization of tracking systems by FIFA [24], many other technologies are also taking part into soccer player tracking, including different accessories into the players' uniforms, one example is the use of GPS devices placed into the players' shirts [34], [88]. However, optical players tracking has the advantage of not being intrusive on the players and that it requires just the video cameras and not any other support accessory, it's cheaper and the methods can be exported to other areas on optical tracking. More recently even artificial

Figure 1.3: Match Analysis system flow. Source: `http://matchanalysis.com/process.htm`

intelligence systems are integrated into tracking players.

Due to the nature of private commercial software, it is not clear in general terms if the implementation is fully automated and which algorithms or methods are used to perform the object tracking tasks.

The Laboratory of Pattern Recognition and Intelligent Systems (PRIS-Lab) has developed a tool to perform Automated Soccer Players Tracking. This platform, named ACE, processes off-line video recordings of soccer matches' obtained from a single point of view, and returns the *tracklets* of the players' positions throughout the video. The current project development is on this platform, accessing the source code and soccer matches ground truth data base. The main goal is to improve robustness of the platform to occlusion events by implementing a Multiple Hypothesis method on it's multiple object tracking process.

1.2 Document Outline

The present work is organized as follows: first and current chapter is an introduction to object tracking and automatic soccer player tracking to define the field of research and the problem to solve, it develops the State of the Art i.e. the study of related professional literature to settle down the line of work, and the scope and objectives of the project; chapter 2 contains the theoretical background on optical object tracking and multiple hypothesis approaches; chapter 3 corresponds to the methodology, where framework and time frame are described; chapter 4 covers the Multiple Hypothesis with Multipartite Graphs implementation details for multiple object tracking as well as other developments into increasing the tracker performance, chapter 5 presents and discusses the results, chapter **??** includes final

conclusions and recommendations. The final section contains the appendices with different supporting documentation, as well as an additional brief research on estimating players' distribution on the field to dynamically enable/disable tracking algorithms in a multi-feature tracking platform.

1.3 Professional Literature

There exist different methods and algorithms that have been developed to solve the many problems that arise when trying to track multiple objects, specially when these objects share so many features in common. M. Villalta [92], based on the Object Tracking survey by A. Yilmaz et al. [99], proposes a classification of these methods in three main classes according to the core functionality of the tracking algorithms, suited as Deterministic Methods, Probabilistic Methods and Data Association Methods. It is important to highlight that the order of appearance of these classes is a remark on the timeline of evolution of the approach mindset, as prior methods were not able to completely solve the Multiple Object Tracking (MOT) problem, it also depicts the focus transition when simpler issues were solved and the more complex had to be taken into account, like occlusionevents. Furthermore, M. Manafifard et al. [49] bring a detailed and extense survey on Player Tracking in soccer videos, reviewing about 163 different publications on the issue, studying cases like Multiple Hypothesis Tracking (MHT), Particle Filters, Graph Representation and Occlusion Detection and Handling. From this study it is important to highlight the following: detection (segmentation, background substraction) is critical in the performance of the tracker, occlusionsare the biggest challenge in MOT, MHT has improved different existent methods in occlusionsolving matters outperforming single hypothesis tracking, there is no mention of Multipartite Graph (MPG) approach as presented by [92] nor a mixture of MHT and MPG resulting in Multiple Multipartite Graphs (MMPG).

The following reviewed literature is used to illustrate the previous statements, the evolution of different approaches and to sustain MHT for occlusionmanaging. In addition, a subsection has been separated to discuss the works which implementations are focused on being robust to identity ambiguities and occlusionevents, appendix **??** depicts the distribution of reviewed methods in a table.

Deterministic Methods

These are methods that use visual characteristics to perform a deterministic trajectory tracking, i.e. there is a model that describes its motion and/or its appearance. The basis of most of these methods is to use brute force to find areas on an image that match a predefined and well known pattern. These methods fail as the track is lost when objects interact with others alike (players of the same team) and can't solve occlusionby itself.

The main approach in this category is known as template matching, where a predefined template is compared with different regions of the image, whenever it matches best, an object (player) would

be identified [10], [79]. The problem with template matching is that the algorithm is fully dependent on its similarity with the objects of interest, if the template is not compatible with the players shape or color in a determined frame, it will fail. This is common on soccer player tracking due to the many visual changes a person has as it moves around the field turning around, crouching, falling and other shape shifting events, enforcing the necessity of a big and robust templates dataset to consider the many scenarios, also, the tracking result will have Identity Swap as the players of the same team have similar chromatic distribution.

Lefèvre introduces the use of active contours into soccer player tracking with a snake-based method [46]. The non-rigid nature of a soccer player is considered in this method, allowing a reconstruction of the contours of the players whenever its necessary, adding flexibility in comparison with a fixed template-matching. However, it would fail when occlusionhappens, as the contour may consider both players (in a 2 objects occlusioncase) as a single player, due to one player blocking the other in a 2D view.

To combat the occlusionproblem, some methods implement data fusion of information provided from different sources to aid the players detection. A template is updated with newer information from new observations over time, characteristics as size, texture and color are used to differentiate between players, and states are stored so that correct assignment is done after the occlusionpasses, which doesn't fully solve occlusionassignment. This method increases the complexity of the tracking platform as it incorporates an observation unit to update the templates, and also the template complexity by adding new features to recognize (color, size, texture). As it still depends on a template, the right update of the template and its initial state are crucial, which forces to manually declare the initial states of the match to aid tracking [53], anyways, similar and occluded players remains a problem.

Probabilistic Methods

Most of these methods are based on Bayesian inference and estimations to perform object tracking. They were introduced as a solution to optimal estimation of non-linear non-Gaussian state-space models, situations where the typical linear model-based Kalman Filter was not enough [22]. The main idea of these methods is to estimate the position of the object on a frame and update its likely position with new measurements, the measurements are assigned certain probability (weights) based on previous information until an optimal result is selected. It is important to highlight that these methods' decisions are based on probability, it chooses the most likely solution based on the cumulative probability of the

assignments along the different frames, therefore, a wrong assignment will spread over time.

One of the most popular probabilistic algorithms for object tracking is known as Particle Filter (PF), introduced in 1993 as bootstrap filter, a numerical method for the solution of optimal sequential state estimation problems in non-linear non-Gaussian scenarios [57], and with increasing applications due to how the algorithm has evolved in hand with computational power. Due to its extended use in the topic it has been considered important to dedicate this section to its implementation.

Czyz, Ristic & Macq implement a color-based particle filtering algorithm, it was first designed to work with single object tracking, thus leading to wrap all players of the same team in a single global state. As particles are attracted to areas of high density, error from a single player is dragged through states update. The solution approached is to add a color histogram and to join detection and tracking tasks to distinguish players of the same team [19]. The experiments show that PF is a powerful tool on MOT, but the scenes analyzed did not consider occlusionevents, and the issue of particles moving towards areas of high density persists, which leads to Identity Swap of players that are close together. To solve the wrong assignment due to particle mixing, some authors have implemented single particle filtering for each player.

Single particle filters per player increases the computational processing requirements as the algorithm is applied for each player. Dearden, Demiris & Grau not only use Particle Filter for each player but decide to spatially segment the most likely areas where the player may move, thus including a spatial window and delimiting the possible solutions of the algorithm; PF is then applied on the delimited area [20]. This approach diminishes the influence of one player's error into the rest, however, the occlusionproblem remains. During an occlusion, the particles used may mix and get messy, therefore, the reliability of the data obtained throughout the event decreases the longer the players remain occluded. After the players separate from each other, using color-based recognition, the particles are reassigned to the respective player by estimating the trajectory that each player had before the event. Even though, the authors mention that particles mixing is still a possible outcome, so that all particles are assigned to a single player (therefore, losing the other player) or leading to Identity Swap with players of the same team.

Kristan et al. implement multiple single-trackers to confront the multi-target tracking problem, including closed-world assumptions to aid in the visuals processing [43]. The robustness of the algorithm is improved due to the constraints of a closed-world, as it was tested for indoor matches, however, those assumptions cannot be considered when working with outdoor sports, as outdoor conditions are

not easy predictable (e.g. a cloud's shadow may cover the field totally or partially for a short or long period of time, it may rain or get foggy), also, the camera is statically placed above the court hanging from the ceiling, position that cannot be achieved in an outdoors stadium.

A major problem when dealing with Particle Filters is the degradation of particle assignment, specially when dealing with occlusion, as particles are wrongly assigned due to lose of information in the involved frames, which leads to losing players' identity after an occlusionevent (Identity Swap and new player condition after a split), therefore, PF alone cannot solve the problem. Itoh et al. [35] introduce a time-situation graph beforehand to an occlusionevent, the graph is used to solve the players' tracking lost after an occlusion, the quantity of players involved in the event is derived from the graph association prior and after the players interaction, thus allowing to assign the "new" players detected after the occlusionto the existing ones after the occlusion, fixing the splitting problem, but the swapping remains as the algorithm is based on PF. Similar approaches introduce the use of graphs to retain the states of the players so that they are not lost after an occlusionor in boundaries conditions as when player exits the frame (for broadcast recordings) or exits the field limits.

Data Association Methods

These methods address the object tracking as to find the optimal solution to a data association problem, where the data to associate is obtained from the object detection step. The main goal of these methods is to solve the wrong assignment and object track losing due to occlusionor similar players collision, most of the time used as a complement to a probabilistic/deterministic approach.

The simplest implementation of a data association method is using the Kalman filter due to its recursive nature, used to update the object-track assignments [75]. It is stated that for multiple-object tracking, there should be an additional computing to identify each detected object before actually tracking them, thus the method results are dependent on the right object detection step frame to frame.

The preferred representation of data for these methods is via a graph, were usually the nodes represent different states of a single player or team, and edges are weights that connect one state with the possible next. Soomro, K. et al. [86] implement a graph approach where each team is represented by a graph, in which the nodes correspond to player positions and the edge weights depend on spatial inter-player distance. They use color-based tracking to differentiate between teams to assign the player to one or another graph, and teams formations data to aid into the estimation of the position of the

players, with the assumption that players remain in an organized manner, therefore, it is expected that error increases when players won't completely follow the initial formation. It also uses only video clips from broadcast matches with an overhead view, occlusionis dismissed.

Figueroa et al. use the graph representation for player tracking by focusing on the occlusionevents, where the nodes are used to represent different blobs to distinguish the players, aided with features as mentioned in above studies, like multi-camera view and color. Moreover, the study adds the use of morphological features for blob separation (like a pixeled player's size and shape model); the algorithm uses a backward and forward graph representation to optimize the graph and aid on multiple cameras to simplify the graph in occlusionevents. This is done by selecting the best view depending on the distance of the blob to the center of the field of camera view and which one brings the most information of isolation of a player, i.e. if in a view the occlusionoccurs, other camera can be used to verify if a player becomes isolated in that view [25], [26].

Wei-Lwun et al. use the graph representation to keep track of the previous assignments, using Bipartite Graphs (BPGs) to guarantee that one-to-one assignment is done by graph constrains; the graphs allow continuity from existing tracks, in contrast with other methods which nature tends to the creation of new tracks anytime an object is detected (as with particle filtering after occlusionwhen dealing with MOT). Wei's implementation considers the players as entities, each containing a set of features (faces, numbers on uniform, skin or hair color) that compose the Conditional Random Field (CRF), thus integrating different sources of information to improve the player recognition and tracking [96]. Siles, F. follows a similar approach by combining a set of features (color, texture, shape and kinematic model of soccer players) to the edges of a BPG to aid the tracklet-player assignment [81]. Later, Villalta, M. expanded the graph usage from a BPG to a MPG, the assignment is the solution of a graph energy minimization problem [92]. Villalta also includes a single panoramic view of the whole field by stitching the images from two 4K cameras used to record the matches. This approach eliminated problems related to resolution or temporal segmentation needed for clips from TV broadcast, however, occlusionproblems remain as the optical capture is from a single point of view, even though the use of MPGs improves the correct assignment of blobs identities, occlusionis not fully solved. It is also important to highlight that using two 4K images recording impacts in the amount of data to process (2x4k cameras of 3840x2160 px each, recording at 30 fps, leading to a load of 995 328 000 pixels per frame), thus, the use of high capacity computers like clusters or parallelization of the algorithm is needed to speed-up the process [93].

The typical implementation of a graph tracker requires to divide the whole video track into several blocks to process, where the initial state of the next block is defined by the graph solved from the previous one. There are two considerations on this: first, an error on any assignment (new blobs, Identity Swap, etc) will spread to the next block, second, the graph solving is a local solution for the block being contemplated, there exists information that's ignored on the current block because it is contained in further blocks, specially in occlusionmatters. Gedikli, S. et al. [30] implement a probabilistic method for estimating the trajectory of the players adding a MHT algorithm. The MHT preserves multiple hypotheses and their evolution through a defined temporal sequence (window), at the end of the window a decision is made with the best fitting hypothesis; Gedikli states that MHT was able solve some misclassifications result from occlusionevents that the typical tracker could not solve.

Multiple Hypothesis, initially presented as a solution for noisy and complex object tracking scenes in 1979 by Reid [1], is formerly widely used in scanner applications [6], where different measures are received on each scan cycle and spreading of a wrong assignment is critical in application such as projectile tracking, submarine location and others with a tight error acceptance. Moreover, it is recently extended to scenes considered complex due to the dynamic interaction between the objects to be tracked, like pedestrians and sport players tracking, where it outperforms single hypothesis algorithms [49], [100]. Due to its high computational cost, it is usually used as a complementary aid for other algorithms, commonly particle filters and/or a graph representation, instead of being used as the core algorithm of the tracker. Breitenstein's [14] experiments show that adding a MHT layer improves the results from a single algorithms implementation, in this case using MHT as an addition to PF, in particular in soccer with a Multiple Object Tracking Accuracy (MOTA) higher than 80% for the scenes selected for evaluation, however, identity switches persist due to the use of particle filters and robustness is low during the initial sequences.

Therefore, most improvements on tracking results due to robustness on occlusion events are found from a mixture of algorithms, usually by mixing MHT with other algorithms (Kalman filter, PF,) or similar approaches like a multi-layer outline, or, with PF based algorithms [9], [58], [101], where MHT has proven to be comparable to "stat-of-the-art methods on standard benchmark datasets" even in its early 90's implementation [40], [49].

Robustness against identity ambiguity

occlusionevent, as stated before with different examples, is one of the biggest problems in optical object tracking. Of the several ways to fight it, some examples are: improving the current algorithm with more features, omit occlusionand care only for previous and posterior identities, increase points of view adding more cameras, and treat the tracking as an optimization problem via multiple layers of assignment (typically via MHT or graphs plus additional criteria).

Hess & Fern improve the PF with their Discriminatively Trained Particle Filters [32], which is an alternative for the typical particle filtering approach by training the data set of the players with features as appearance (size, color), motion (via 11 different probabilistic features) and player interaction (to prevent particles mixing between different players), all features constantly updated frame to frame. This method increases the complexity of implementation of the algorithm as it requires to update the features and train the data set for each player, but shows great robustness when occlusionoccurs, especially considering that the algorithm was implemented in American Football, sport with more contact between players than Association Football.

Ok, Seo & Hong attack the particles mixing problem when using PFs by implementing an Occlusion Alarm Probability (OAP) [60], intended to repel the particles from sticking between players in multi-target tracking, thus preventing particles mixing and Identity Swap. The results show that there are some occlusioncases in which the repel leads to unassigned particles to one of the players, after the occlusionends, the player previously hidden is now considered as a new player (blob) entering the game due to the missing information from its previous states during the occlusion, this due to not considering tracking splits scenarios.

Sabirin et al. [74] detect early occlusionevents by defining a rectangular Region of Interest (ROI) to separate each player, rather than a tight contour, when two or more of these regions are overlapped by a 60% of their area, an occlusion is detected. In general, occluded players are considered a big single ROI during the occlusion event, their identities are assigned after the event, when there's enough information to identify the players during the occlusion, single regions are assigned preserving the shapes from previous frames, updated just after the event finishes. There's no complete identity solution during an occlusionevent, making the algorithm robust to occlusions (the identities are recovered after the event) but not occlusionsolving. It is important to highlight that Sabirin included players' texture as a feature for object identification, alongside other features as position and estimated motion based on optical flow in the hue channel, to identify different players, showing it as an useful feature allowing easily

differentiation from different teams and also being the key on the algorithms robustness to occlusion. However, the method is dependent on the segmentation results, as many others, thus the importance of efforts on improving the frames segmentation.

Morais and Xu [55], [98], although working with the indoor sport futsal, both suggest the use of multiple cameras to improve the players distinction, as different planes will add position information that a single plane cannot easily obtain, the positions of the players are estimated using homography. Iwase, S. & Saito, H. implement multiple view images in outdoors scenarios locating cameras around the game field [36]. Even though the use of multiple cameras has proven to improve the segmentation results and diminish the occlusionissues (especially when the field is surrounded by cameras and 3D information is obtained), it is dependent on the optimal different views needed to prevent occlusion, which can anyway happen for non-rigid 3D objects in motion, e.g. one or more players may fall over a single player so that it is completely hidden from any camera, or there may be a group of more than two players colliding, which leads to players ambiguity associations. [42] also mixes information from multiple camera views into an MHT environment, using 3D information to aid correct identification of players during an occlusionevent. To prevent occlusionusing multiple cameras, new research questions may arise like: how many cameras? And, where to place them to cover the most, if not all, the 3D space? Thus, making the solution hardware dependant. In [49], the previous questions are exemplified by a variety of camera configurations, many different ways to place the camera and occlusion persists being a nuisance.

Jiang [38] represents the possible locations of a single object as nodes of a graph, building a multi-layer graph with the occupation probability of the object, a set of graphs would represent the multiple objects to track. Shitrit et al. [80], in a similar way, implement a graph representation for players' tracklets, reducing the problem to a multi-commodity network flow optimization, using K-shortest path to solve the assignments on a multi-layer graph. These representations work similar to a MHT approach, where different tracks are preserved and optimized to find the best later on, instead of picking local bests on each iteration of the algorithms. The above implementations resemblance a MPG with MHT approach, differing in the way the graph is constructed, which is considered on the current work's implementation as a MMPG.

In [50], it is used an appearance-based model MHT, in which they consider appearance and motion cues to refine players' identification. The combination of different features in order to improve the segmentation, as stated by Manafifard, is a needed modification to data association, as these

methods usually start by considering that the nodes to associate (in a graph representation) are the right measurements. They show that an appearance-based model MHT performs better than a simple MHT algorithm, giving it the a better discrimination of players and false alarms. This consideration is taken into account in the current project, where the nodes of the graph are not only a label and a position, but a set of optical features that weight on the association solution phase.

MHT outperforms those methods using single hypothesis in the case of occlusion and noise [49], however, it is still not a flawless method, as it depends on the estimates of the measurements within an occluding blob, where a sudden change on velocity (reality differs highly from estimation) or a simple bad estimate results in tracking error [39]. Also, it has a high computational cost that makes it unattractive for real-time applications, nonetheless, it sets an interesting paradigm from which a multi-layer structure for assignment solving can be implemented [38], [80], which sets a different way to resolve the typical graph optimization problem. The previous leads to the idea of implementing a multi-hypothesis/multi-layer structure as a mixture of MHT and MPG, a Multiple Multipartite Graphs (MMPG) tracker.

Artificial Intelligence

More recently, with the approvals and standardization in the use of performance and tracking tools in sports like soccer [23], [24] and the raise of Artificial Intelligence (AI), AI powered tools have been incorporated to aid in player's performance evaluations via automated optical tracking, from the generation of ground truth data sets to train, test and validate the tracking solutions [61], to the integration of AI to track the players real-time from soccer matches recordings, detecting off-sides, goals and tracing players' trajectories [2], [4], [66], [67], [78], [85].

However, it is important to highlight these solutions do not rely solely on artificial intelligence, they follow the thread of combining different tools and methods into a robust and more complete platform, including but not limited to wearable trackers, installation of multiple cameras mounted underneath the roof of the stadiums ([78] use 12 dedicated tracking cameras) and human aid to monitor the results and make corrections as needed, since even with AI errors can occur, as recorded on 2020 when an AI camera operator confused the head of a bald referee as the soccer ball [3].

1.4 Problem Description

Research question

Will a Multiple Multipartite Graphs approach improve precision of soccer players' tracking in comparison to the current method based on single Multipartite Graphs?

Problem

Automatic multiple object tracking is not a flawless process, there are different events that make it a difficult task, in particular, the occlusion events. When performing optical object tracking, occlusion events cause information losing that leads to wrong or incomplete tracking results. Many algorithms have been implemented to solve this issue, but they fail one way or another, making it necessary to add a final layer of human aided correction, thus current tracking platforms are not fully automatic.

1.5 Hypothesis

By implementing a Multiple Multipartite Graphs data association tracking method, it is possible to improve the occlusion event management of the tracker, which will lead to a metric value of MOTA greater than 75% on a whole match.

1.6 Objectives

General Objective

Implement a Multiple Multipartite Graphs occlusion solver to manage occlusion events on automatic soccer players' tracking from Ultra High Definition video.

Specific Objectives

- Develop the framework needed for the Multiple Multipartite Graphs tracker.
- Design a metric to detect simple occlusion events on soccer matches.
- Implement Multiple Multipartite Graphs occlusion solver.
- Validate the algorithm against annotated ground truth datasets.
- Disseminate research to scientific community.

1.7 Scope

The current project focuses on the occlusion events problematic, it departs from the previous ACE platform version which includes multipartite graphs to perform players' tracking, using panoramic field view in ultra high definition (UHD) images from the stitching of two 4k recordings of each half of the soccer field. The platform will transform from using single multipartite graphs to a Multiple Hypothesis Tracking scheme, self-named Multiple Multipartite Graphs. Some other changes are considered to improve the platform, according to different specific needs of the project encountered during its implementation.

Chapter 2

Theoretical Background

This chapter covers important concepts and key theory regarding Multiple Object Tracking (MOT), Occlusion, graph representation and the intuition on Multiple Multipartite Graphs (MMPG), as well other concepts related to complementary integrations to ACE tracking platform with the goal of improving the tracking results.

2.1 Multiple Object Tracking - Soccer Players Tracking

To perform soccer players tracking, a single player must be distinguishable from any other player along the match. This task, as easy as it may seem for the trained eye, is a hard task in object tracking, taking into account the next considerations: not relevant objects (e.g. goal posts, banners, audience) must be filtered from the objects of interest (players, ball, referee), each player must be identified in each single frame, the same player must be identified throughout the video (consecution of frames), the referee is not a player (can be or not removed from the *tracklets* depending on the desired analysis), there are active players substitutions during the match with fresh players, both teams fight for the same ball leading to occlusion and tons of players interaction, and many others. To cover all of these and more, object tracking is separated into different steps dedicated to specific tracking tasks.

Three key steps can be recognized: detection of objects of interest, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior. The whole tracking task can be defined as "the problem of estimating the trajectory of an object in the image plane as it moves around a scene" [99]. Tracking objects can be messy due to loss of information caused by projecting the 3D world onto a 2D image (some platforms like SportVU [90] operate directly with 3D data from multiple views coupling), noise in images, resolution of the video recordings, complex object motion, scene illumination changes, partial and full object occlusions, and other problems application dependent; each step has its own difficulties.

The three key steps mentioned above can be recognized in the implementation of the computational

platform ACE. The Pattern Recognition and Intelligent Systems Laboratory (PRIS-Lab) develops a computational platform for soccer digital video analysis, called ACE, platform with the task of providing automated soccer matches analysis to human motion scientist, coaches, teams and general public. This platform is implemented in a multi-layer architecture comprised of two main stages: perception and cognitive [82].

First stage is covered by the temporal and spatial segmentation blocks. The temporal segmentation performs detection and classification of scenes with players information (applicable to videos of broadcasted soccer games) to remove those recording scenes that have no information of interest, like advertising spaces; if the soccer game is recorded in panoramic view of the whole field, this block is not needed. The spatial segmentation detects and localizes the objects of interest in each frame of the scene.

The second stage is covered by the trajectory tracking and semantics of the game's tracked activity. The trajectory tracking is where the position-object assignment is performed, using as input data the results from object segmentation. The semantic analysis performs a reinterpretation of the trajectories to detect actions and classify events, this block is responsible of generating statistics, occupational map and other information to provide to the customer [81].

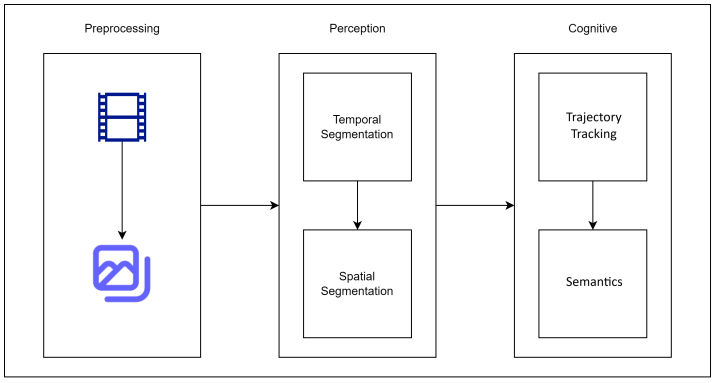


Figure 2.1: ACE computational platform architecture layers

Different methods and algorithms are used to try to solve each object tracking step. As the project focuses on the players trajectory tracking, the current scope prioritizes methods and algorithms directly related to the trajectory-object association. The inputs of the algorithms to study are the objects spatially segmented by the previous block in ACE's flow, i.e. **player blobs**, block which graphically identifies them by drawing a contour around the player's detected shape, the centroid of the contour

is used as the estimated position of the player. It is important to highlight the influence of the results of the segmentation task into the trajectory tracking, any error in the players' segmentation and detection will highly lead to a wrongful tracking. Details of the base implementation of ACE platform are discussed in [92].

To understand the complexity of assigning the positions detected in a set of frames to a player and identify its trajectory, it is important to know how a typical soccer game is broken down. First, the game officially starts with all 22 players "easily" distinguished as they are separated one another following a game strategy formation, both teams start located in opposite sides of the field. As the game begins, the players start to move over the field reacting to the game events as the match unfolds, a player may decide to stay in position, move arms or crouch (shape change), jump, go forward or backward, left or right, follow a player or the ball; as the ball moves from one side of the field to the other, the player may change its previous intended trajectory to another; the motion is errant and not much predictable without the semantics of the game, even though there is a general strategic plan. If the players would not interact one with another, the trajectory-player assignment may not be as hard, but, as the game definition states, both teams are fighting for the same ball, this leads to more than one player sharing a close range to the ball as they struggle to control it, possibly generating occlusion when one player is placed behind another (from the perspective of the 2D image recorded from the same view plane); which player is which? Using the colors of the soccer suits is useful only if the players are of different teams, but if the players are of the same it won't help much, even less if it is confused with the background (see Figure 2.2).

Figure 2.2: Players detection issues due to visual properties, scene conditions and segmentation process. Results from ACE platform, 2018

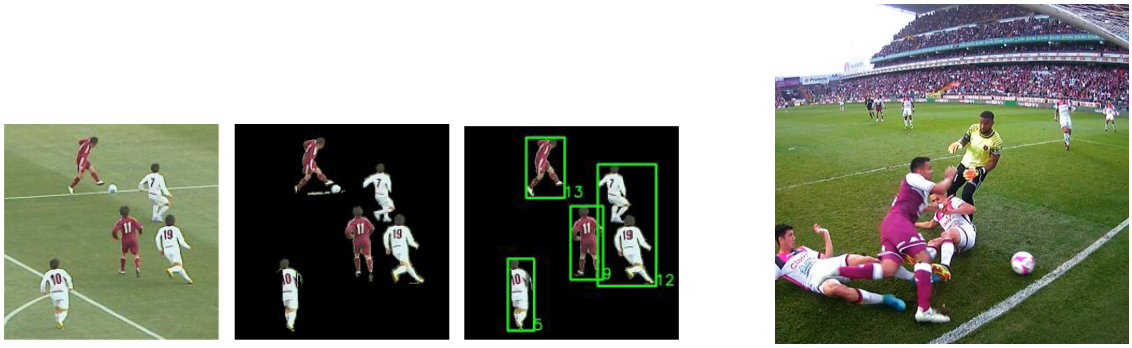
Automatic object tracking has to deal with previous mentioned issues and many others. Most of

these issues correspond to the first stage, specifically from spatial segmentation, as they are dependent on the image characteristics. A perfect segmentation or feature extraction would simplify the second stage, where the association algorithms could operate with clean and organized information. However, as these issues are innate to image processing, data association and object identification have to deal with messy and flawed detected objects. Even though there are different methods that try to improve the quality of the detection phase results, some events persist to the other phases, like occlusion events, resulting in the identity assignment stage consuming data with those problems, causing the previously mentioned errors like Identity Swap, missing identities and spurious objects.

2.2 Occlusion

When using images, the features extracted depict visual characteristics of the scene under evaluation, one way or another, they describe each object contained in the scene. However, images are but a projection of a 3D world into a 2D plane, where these visual characteristics are transformed and even hidden, making them hard to extract sometimes. Typical features extracted in this context relay on pixel color (using histograms of color in different representations like YUV, RGB or HSV) to detect areas of interest in the image (background extraction, possible objects detection) and shape/size of the objects to filter spurious detections. However, some information is lost in the 3D to 2D world transformation, sometimes vital information like depth location of the objects, which leads to occlusion events.

In an occlusion event, objects that share a common axis location, but different depth from the camera, appear to share the same spatial location, as depth information is lost in the plane of the image. When this occurs, it is hard to identify which object is which, as figures and colors are mixed and the common segmentation methods fail to distinguish each object involved; this particular scenario, as stated in [49], is the biggest challenge in MOT. Figure 2.3 illustrates occlusion events on real soccer matches, even though two close players are not visually blocked for the human eye, its proximity and visual features can lead either segmentation or tracking algorithms to give a faulty result (figure 2.3a). Some other events are more complex and practically impossible to automatically correct (figure 2.3b).



(a) Minimum proximity between players is considered as an occlusion event by the algorithm, which groups both players as a single one. [74]

(b) Four players occlusion on TV broadcast video.

Figure 2.3: Examples of scenarios perceived as occlusion events.

The existence of occlusion events, and a segmentation process that is easily affected by it and many other noisy situations, makes it a need for the identity assignment phase to be robust in order to deliver a clean tracking output, it must deal with false detections and missing information. However, such a perfectly robust algorithm does not currently exist, and the current ones which outputs have the higher accuracy percentages are not fully reliable on its own. Recent approaches tend to mix different methods to aid the core algorithm into solving assignments, be it hardware solutions like increasing the points of view, or a mixture of feature models and tools [45], [49]. In terms of occlusion , three ways to *confront* the event were identified: no-handling of the event, event detection, and event solving.

No-handling implies that occlusion is not considered as an input event for the identity assignment. These methods usually rely on the ability of the algorithm to make corrections or its robustness to false detections. As discussed previously, there is no flawless algorithm, and this approach will lead to an incomplete tracking history that requires human aided cleaning of the results. Other way of not handling the occlusion when tracking is trying to prevent it from happening when capturing the scene, this can be done by increasing the points of view to perform a 3D projection of the field. With depth information, most occlusion scenarios are dismissed in exchange of having to care about homography and camera placement. However, even in a 3D projection some occlusion can prove to be challenging (like in figure 2.3b). More recently, with advancements on technology and its usage approval in official games [23], [24], tracking is aided, if not even replaced, by alternatives to optical image segmentation, the use of GPS trackers provides an accurate occlusion-free extraction of the players' position at any point of the game with a highest refresh rate than that provided by 30 or 60 FPS videos, simplifying the tracking challenge to matching the GPS tracked position to coordinate in an image [88].

Event detection is the ability to recognize occlusion events, usually indirectly by human interpretation of tracking information, but to decide to dismiss it, i.e. identities are assigned before and after the frames in which the occlusion occurs, but during the event there is no verification of the identities. This can be seen in [25], [26], where the missing nodes in the sequential graph hints an occlusion event (the graph passes from having two identified players to only one during many frames, and then again having two players, see figure 4.1). However, there is no occlusion handling, the missing information is not corrected and the tracking output only shows the player missing in some frames, but it is considered enough to know the players' position before and after the occlusion , as sometimes the events are only a few frames long, in the best of cases. However, scenes are not always so simple and there occurs identity swapping and false-new-player labeling after the occlusion event.

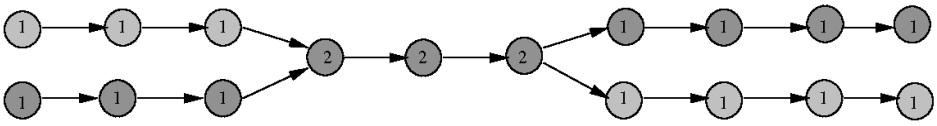


Figure 2.4: Sequential graph where missing identities suggest an occlusion event [26]

Some occlusion detection techniques are used to improve the main algorithm behavior, like in [60] where occlusions are detected by a proximity alarm in order to disperse the particles from the Particle Filter (PF) that is used to track players. Another example is [74], that uses occlusion detection to trigger occlusion handling.

Occlusion handling refers to trying to solve identity assignment during an occlusion event, and/or to verify and correct identity assignment before and after the occlusion occurs. Some examples are [25], that builds forward and backward graphs of the frames with missing information in order to merge data prior and after the occlusion occurs; [74] uses different player attributes (like color and movement direction) and prior-occlusion Region of Interest (ROI) placement to define the position of missing players during an occlusion event.

Handling and solving occlusion requires at least three phases: detection of the event (in which frame it starts and when it ends), revealing hidden players, and identity preservation during the event. It is important to note that this is a simplification of what it would take to solve occlusion , as there are different degrees of complexity in which it can occur depending on the number of players involved and if there are other dynamics like players intermittently occluded while others are permanently occluded.

Occlusion event classification and identification

Occlusion events are different and vary in complexity. Simple events as a short sequence of two players can be solved in most cases, for example with a forward-backward sequential graph [25] or with PF [32], however, even in these scenarios occlusion solving effectivity is not 100% and there exists many other scenarios more complex that are yet to be confronted.

An occlusion event O can be identified by its length in video frames, and its complexity. The length can be represented either by the number of frames or by the initial and final frames k_i, k_f. The Occlusion Event Complexity (OEC) is a score that depicts how complex an occlusion event is in terms of the involved objects' interaction and the occlusion area objects density, where $O^1_{5,30}$ is an occlusion event that occurs from frame $k = 5$ to $k = 30$ and has an Occlusion Event Complexity Score (OECS) $CS = 1$, where the OECS is the product of the Occlusion Dynamics Index (ODI) and its Occlusion Agglomeration Size (OAS), to be detailed later.

To study the event's complexity, it can be categorized by its Dynamics as Steady Occlusion Event (SOE) or Dynamic Occlusion Event (DOE), and by its OAS, represented by the number of players involved. From the dynamics perspective, it considers the interactions of the players during the event. It may happen that during the whole occlusion there were only two players present, both move to posses the ball, fight for it for some frames, and then one takes possession of the ball and runs faster than the other bringing an end to the occlusion . The previous example depicts an event in which the OAS is constant, but it may vary in more complex events.

In the case of an occlusion that starts with two players for a couple of frames and later changes to three players when a new one gets close to aid its teammate, the OAS varies, and thus the complexity of the scene increases. In this case, let $O^{CS}_{5,30}$ be the occlusion event described previously, where $o^{CS1}_{5,10}$ and $o^{CS2}_{11,30}$ are the sub-events that comprise $O^{CS}_{5,30}$ (event starts with two players and ends with three), the OAS is useful to break off a complex event into several simpler ones. However, the event can get even more complex by simply adding intermittency to one of the players.

If we take the previous example and add intermittency, let's say that the third player enters and leaves the Occlusion Event Region (OER) several times, the OEC increases as the tracking algorithm must deal with an intermittent merge/split for several continuous frames, result of the third player interacting with the previously occluded pair. If there are many intermittent players, the complexity of the scene increases even more. Thus, we can identify a SOE as one that has no intermittency, with an Occlusion Dynamics Index Intermittency weight (ODI-I) of 0; and a DOE as one with intermittency,

with an ODI-I of the number of intermittent players.

Other factor to consider for the OEC is if the players interacting are from the same team, as similar color distribution leads to Identity Swap, it is easier to distinguish players from different teams. In this sense, we define an Occlusion Dynamics Index Team weight (ODI-T) of 1 if there are players from different teams, an a ODI-T of 2 otherwise. The final ODI is the summatory of the ODI-I and ODI-T, e.g. an occlusion event with two players of the same team (ODI-T of 2) where one is intermittent (ODI-I of 1) would have an ODI of $DyIx = 3$. To calculate the final OECS of this example, we use its OAS, with a value of $AgIx = 1$ as there are only two players present, and multiply it by the $DyIx$, the OECS of this *simple* event would be $CS = DyIx * AgIx = 3 * 1 = 3$; if we take an scenario with the same OAS but with no intermittency and players from different teams ($DyIx = 1$) the final OECS is $CS = 1$, which is the simplest possible occlusion event (a $CS = 0$ would mean no occlusion is happening).

$$CS = DyIx * AgIx = (ODI - T + ODI_I) * [1 + (x - 2) \cdot \mathbf{1}_{x \neq 2}] \tag{2.1}$$

This project will focus on occlusion events with an OECS $CS = 1, 2$, which are simple occlusion events, cases where $CS >= 3$ will be tested, as those are interesting and challenging scenarios (specially if the high score is due to intermittency), but no specific result is pursued over it. The most common OECS are yet to be obtained from scene labeling on real matches, to be shown in further results.

2.3 The Graph Representation

A *graph* G is a mathematical structure that shows relations between the objects it comprises, it mainly consists of two things, a set $V = V(G)$ whose elements are called *vertices, points,* or *nodes* of G, and a set $E = E(G)$ of unordered pairs of distinct vertices called *edges* of G, such a graph is denoted by $G(V, E)$. If there is an edge $e = \{u, v\}$ (the edge e connects vertices u and v), then vertices u and v are *adjacent* and are the *endpoints* of e [73].

Graphs can be pictured by diagrams where each vertex v in V is represented by a dot or a small circle, and each edge $e = \{v_1, v_2\}$ is represented by a curve which connects its endpoints v_1 and v_2 (see figure 2.5 for a generic example of a graph).

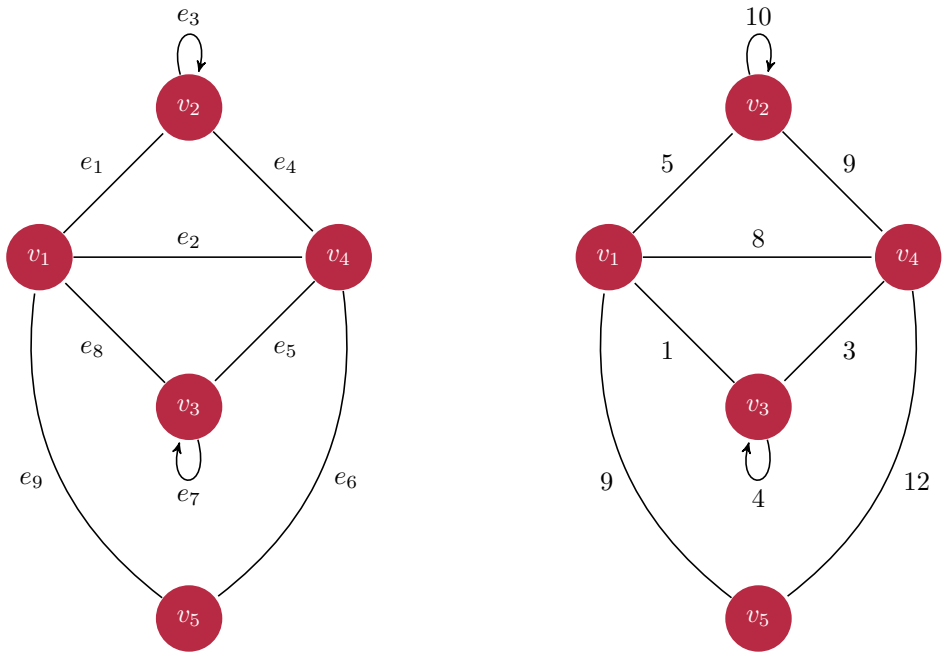


Figure 2.5: Graph G pictured by its vertices and edges relations (Left), example of a weighted graph (Right).

If graph G is assigned data to its edges and/or vertices, it is called a *labeled* graph, in particular, G is called a *weighted* graph if each edge e is assigned a number $w(e)$ called the *weight* or *length* of e, and thus an *edge-weighted* graph (see figure 2.5).

A *complete* graph is a graph G in which every vertex is connected to every other vertex in G (see figure 2.6). A graph is *directed* if it is made up of a set of vertices connected by edges, where the edges have a direction associated with them.

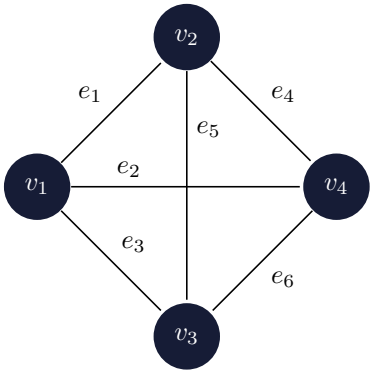


Figure 2.6: Complete graph example

A *multipartite* or *k-partite* graph G is one in which its vertices V can be partitioned into k different independent subsets such that vertices of the same subset are not adjacent. If the amount of subsets

$k = 2$, it is said to be a *bipartite* graph, if $k \geq 3$, it is said to be a *k-partite* graph. It is a *complete k-partite* graph if every pair of graph vertices in the k sets are adjacent, if the subsets have $p, q, ..., r$ vertices respectively, the complete k-partite graph is denoted $K_{p,q,...,r}$ (see figure 2.7).

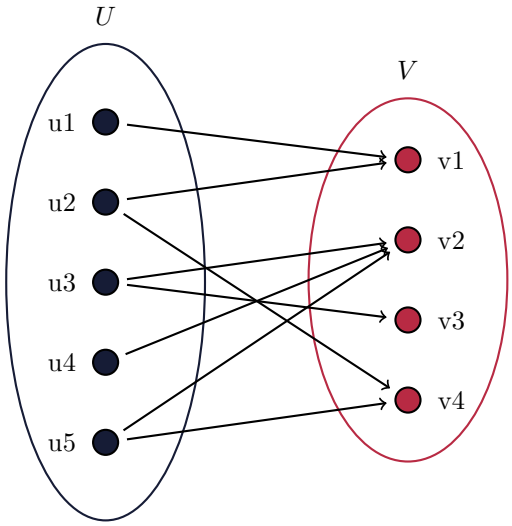


Figure 2.7: Bipartite Graph Example

In the case of ACE platform, a series of k frames are represented by a *directed edge-weighted k-partite* graph *mpg*, where the vertices (nodes) of each subset k represent a set of characteristics for each object of interest (players, ball and referee), and the weighted edges represent a cost of making the nodes adjacent (i.e. the weight of node u_i from subset k-1 representing the same object as node v_j from subset k). The *mpg* is initiated as a *complete* graph, and is then minimized by a shortest-path algorithm to obtain a graph where **ideally** each node has exactly one adjacency (see figure 2.8). If there were no faults when tracking the players, each subset k would have the same exact number of nodes (not considering player expulsions or replacements) and the ideal minimized graph would be the common output, however, that is not the case in a real scenario.

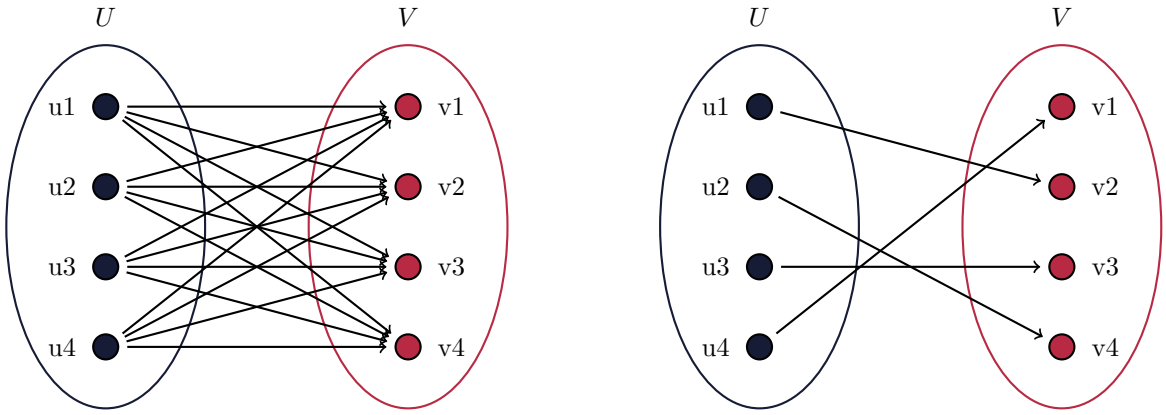


Figure 2.8: Bipartite graph example with 4 nodes in each subset. Left: Complete Bipartite Graph. Right: Minimized Ideal Bipartite Graph.

2.4 Multiple Multipartite Graphs

Object tracking involves detecting and identifying the object of interest in an image k, and preserving its identity in the next images $\{k+1, ..., k+i\}$ from a sequence of consecutive images (detecting and identifying the same object as itself in each image). Different features are extracted to estimate the object's position in the image, in this sense, the identity assignation and position estimation are considered to be an *assignment hypothesis*. Different considerations are taken to determine if that hypothesis actually represents the desired object, in a probabilistic tracking approach, each possible assignment with a high probability is a hypothesis (for example each particle in a PF[22]), where the tracking result can be a single hypothesis with the highest probability, the average of the hypotheses with highest probability, or a region of high probability. In a data association tracking method, in particular a Multipartite Graph (MPG) approach, each node adjacency combination is a hypothesis (see figure 2.9), it can be said that a complete MPG is one that represents the whole hypothesis space.

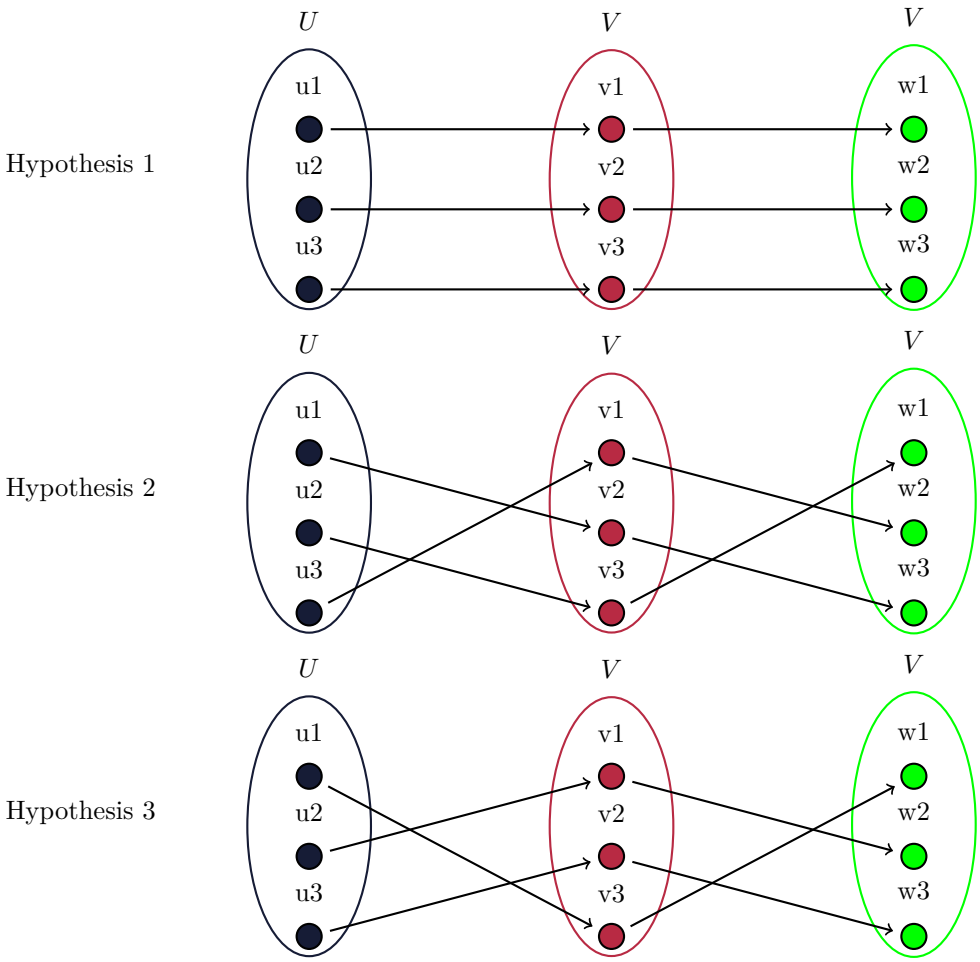


Figure 2.9: 3 assignment hypothesis examples for a multipartite graph of 3 subsets with 3 nodes each.

When considering the features extracted and other constraints (like distance and speed constraints) many of those hypotheses can be easily discarded, as they are *not possible* due to known physical interactions (e.g. a player cannot be on one side of the field in frame k and on the other side of the filed in frame $k+1$, if consecutive frames represent a lapse less than a second). For the *plausible* hypotheses, usually a shortest-path algorithm is used to determine the best fit minimized graph according to the different weights assigned to each edge (from features and constraints). However, the usual approach is to build a MPG for a fixed window size of K frames, and pick the very *best* assignment hypothesis and discard the rest, which would be the final solution if we suppose that the data from which the MPG was built is accurate with no noisy, spurious or missing identities, what happens in real case scenarios, thus the local best hypothesis may not be the global best, or even the best for a MPG that

considers the frames from the same window and the next one.

One way to illustrate it is by comparing the hypothesis decision by shortest-path to what sometimes occurs with a GPS map navigator. It can occur that the driver is suggested to take an exit to go through a path which is just 1 second faster than the current course, the driver may decide not to do so considering that 1 second is not gain enough and that taking the exit would lead to a known hill that the tracker dismisses as it lacks topography information, leading to a slow down and therefore lasting even more than estimated, the algorithm is designed to pick the *shortest* path from the currently available data. It can be argued that incorporating a way to calculate the estimated time of arrival (ETA) considering the hill slow down would improve ETA's accuracy, let's now consider the case in which the navigator does not know beforehand every route from a map, but it generates the best route evaluating a portion of the map by fixed size windows as the car advances, then, until the window with the portion of the map that includes the hill is processed the algorithm won't take into account its existence, it may still offer the driver to continue to the hill, but it is only after the algorithm detects the hill that it can conclude that the current path does not lead to the global optimum. Similarly when tracking football players (specially if the final goal is to track live players), when the current window of frames is being processed, the algorithm does not know how the batch relates to the next, if there is loss of information in the next batch or if the current one is missing a player that is detected in the next; the shortest-path approach picks the local best hypothesis, which may not coincide with a global best hypothesis.

The above scenario may result in cuts in motion flow, for example a motion scene being composed of two MPG due to window sizing and sequential graphs creation instead of a single MPG, an occlusion event could be split into several MPG with its own faulty local optimum that makes no sense once put together; in the end, the final graph representing the whole match is a chain of local bests hypothesis added sequentially, which may not have the same results as a MPG made from the entirety of the match's frames, more study is required to understand the extent of the effect of different window sizes. As of now, the tracking phase is fed an optimum initial state that may or may not include errors and assume it as the truth so as to link each sequential batch (hence the importance of a clean segmentation and detection before the tracking phase to prevent spreading errors the tracking algorithms will take as truths). Figures 2.10 and 2.11 compares the general processing of the MPGs of a scene comprised of 6 frames with a window size 3 and 6, respectively.

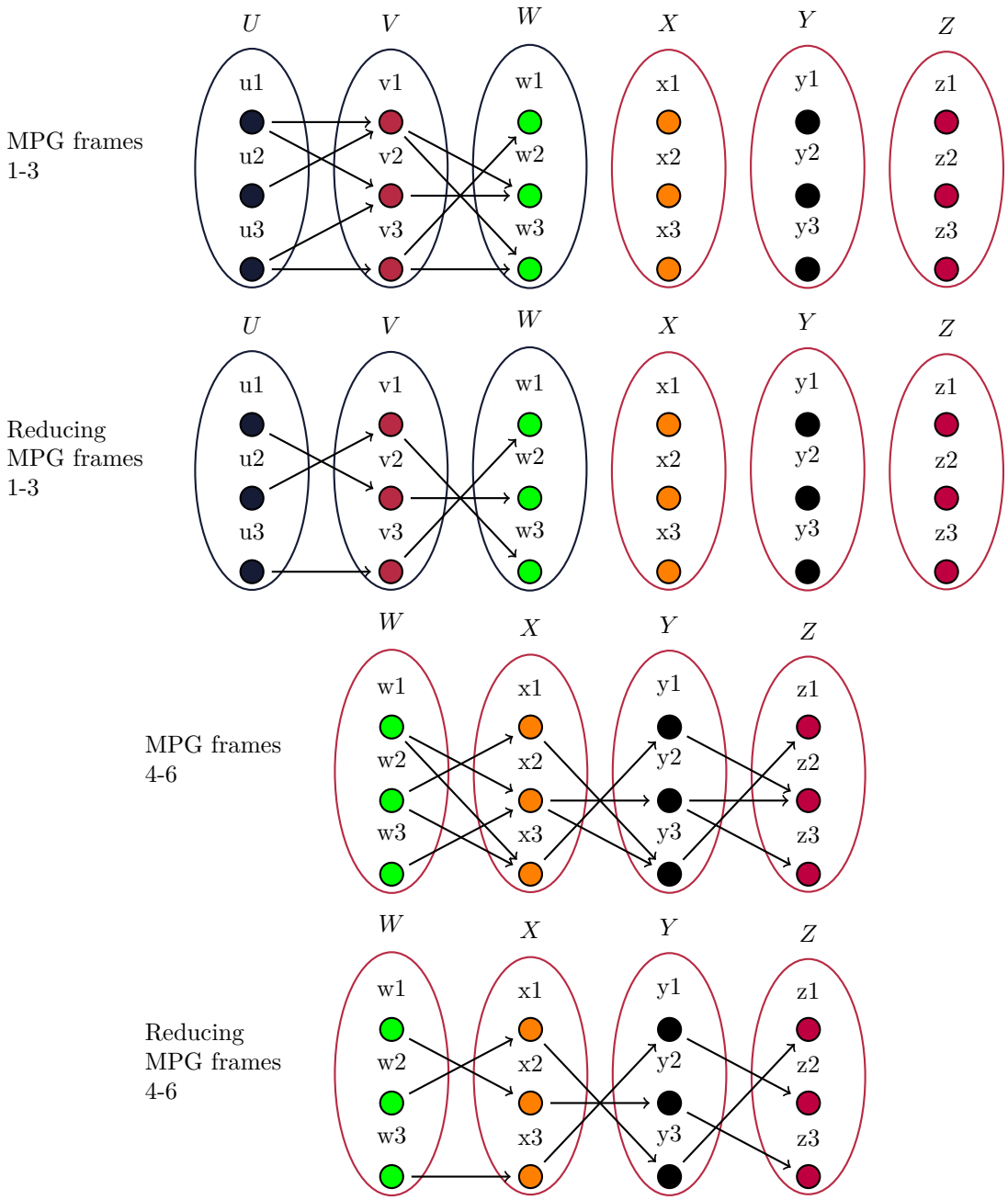


Figure 2.10: Multipartite graph processing with a window size 3 of a scene comprised of 6 frames. Subsets U, V and W represent the first 3 frames of the scene, as the window size is 3, these subsets comprise the first MPG, it is minimized and the final subset W is the starting point for the minimization of the next MPG for the last 3 frames of the scene.

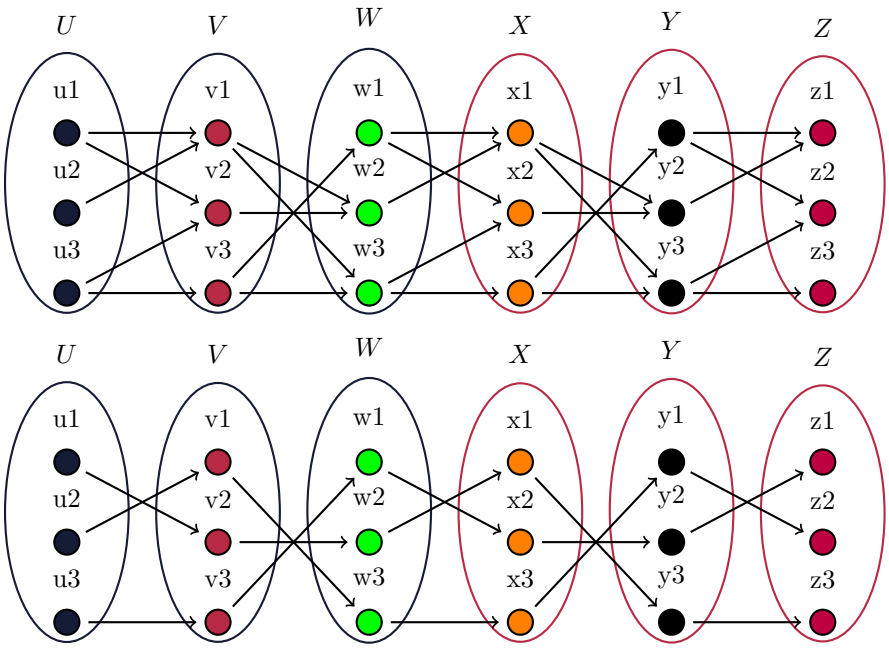


Figure 2.11: Multipartite graph processing with a window size 6 of a scene comprised of 6 frames. Subsets U, V, W, X, Y and Z represent the whole scene in a single MPG, which is then minimized with the whole scene's information.

Considering the local optimum as an assignment hypothesis (one possible assignment solution based on the available data), let's suppose the tracking can be improved by holding the assignment, or hypothesis selection, until there is more information for an actual best pick (we have knowledge that there is a hill coming, so let's be flexible with the MPGs optimization). Keeping a set of best local optimums instead of only the best local optimum grants a set of alternative assignments the tracker can piece together into building a wider optimum without committing to the initial assignment based on limited scene information. The idea of propagating multiple assignments hypothesis was first presented in 1974 by Singer, Sea and Housewright [83] to be used in dense multitarget environments, and later expanded in 1979 by Reid [1], presented as Multiple Hypothesis Tracking (MHT) algorithm, a systematic approach to solve tracking of multiple objects in a complex scenarios. The algorithm mainly consists on creating assignation hypothesis and propagating them as new measurements and detections are incorporated into the assignment space, usually depicted as a decision tree that encompasses the hypothesis space (figure 2.15). Those discarded hypothesis (branches) are *pruned* from the *hypothesis tree*, each new measure adds new information and thus creates a new subset of hypotheses (*hypothesis branches*).

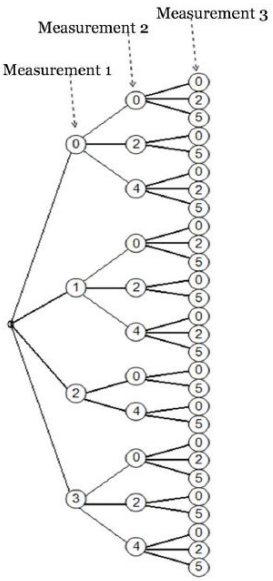
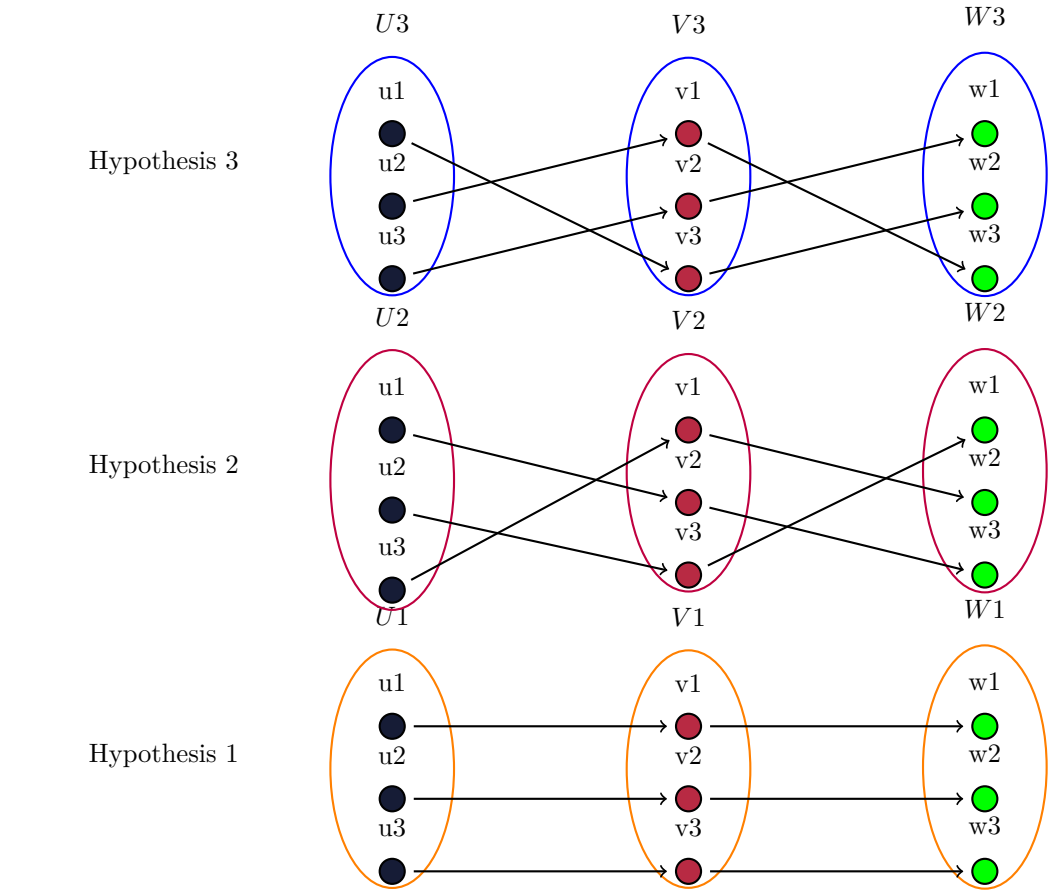
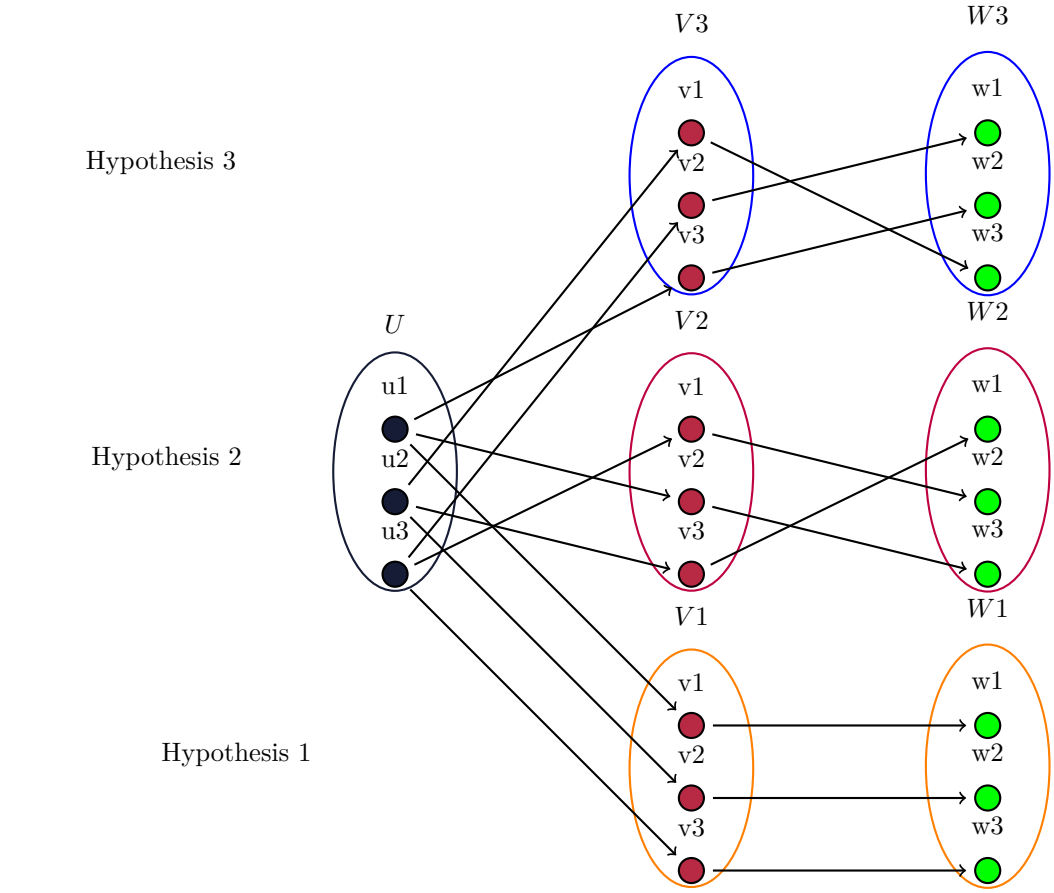


Figure 2.12: Hypothesis Tree: tree representation of formed hypotheses. [6]

There exist different implementations of a MHT algorithm [1], [6], [8], [13], [18], [29], as well as variations depending on the area of research, improvements or the algorithm it is mixed with [14], [30], [42], [50], [58], [101], but two key aspect remains, the first one is its high complexity and computational cost due to the hypothesis creation process (hypothesis can become virtually infinite if not limited) that makes it a requirement to use a robust pruning method and to limit its length with a fixed maximum size. This issue is similar to the one previously discussed with MPG, however, due to its complexity and cost, it is not desired to use MHT for the entirety of the match but rather in sections where it may come handy, like in occlusion events (here it is paramount the idea of occlusion events detection and classification).

The second aspect regarding MHT is its tree-like shape, comparing the main structure of a *Hypothesis Oriented* hypothesis tree [6] with an MPG (figures 2.9 and 2.11), where each branch of the tree is a hypothesis and each possible assignment combination in a MPG is a hypothesis (taking each subset as a single node of a hypothesis branch), the possible MPGs can be arranged each as a branch of a hypothesis tree, thus making the whole hypothesis tree into an MMPG space (figure 2.13), where *hypothesis pruning* would mean to discard an MPG, and *hypothesis propagation* the incorporation of new subsets, i.e. expanding each hypothesis/MPG with new *measurements* from the following video frames. Each new subsequent MPG window adds a new set of *hypothesis branches* to the existent ones, making the *hypothesis tree* grow.


Hypothesis 3
Hypothesis 2
Hypothesis 1
U3
V3
W3
u1
u2
u3
v1
v2
v3
w1
w2
w3
U2
V2
W2
U1
V1
W1


Hypothesis 3
Hypothesis 2
Hypothesis 1
U
V3
W3
u1
u2
u3
v1
v2
v3
w1
w2
w3
V2
W2
V1
W1

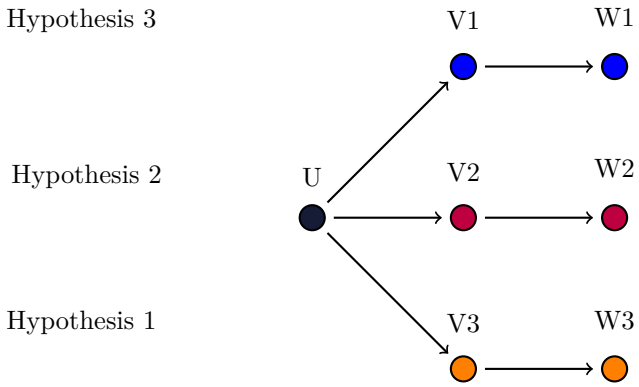


Figure 2.13: Depiction of Multiple Multipartite Graphs structure based on a Hypothesis Tree, where the first common subset is the starting point, each branch is a hypothesis composed of a possible reduced MPG

Following the previous intuition, an MHT-like structure can be obtained by an MPG of MPGs, where each MHT represents a hypothesis branch.

2.5 Player Identification

An essential aspect of a correct object tracking is the ability to retain the unique identity of each object throughout the video. In optical object tracking, characteristics extracted from the video can be used as a first layer of attributes that distinguishes one object from another; properties derived from the first set of attributes, as speed or trajectory direction, can be used as complementary attributes. Moreover, building a set of attributes can be extended to groups of objects to determine belonging to a certain group (spurious blobs vs player blobs, referees vs players, team A vs team B, etc).

Player Model

From a 2D image object identification standpoint, an object of interest is but a cluster of pixels that standout from the rest of pixels in the image. From these pixels some attributes can be extracted to build the object's characteristics that make it a unique object. Directly from each pixel we can extract the color information in different formats, hue, saturation, its position relative to the global frame of reference; from the cluster of pixels we can calculate histograms of visual properties, centroid, contour, area, perimeter, aspect ratio, and more. This first set of attributes becomes useful when identifying objects in the same image scene, as the slightest variance can be used to distinguish between similar objects. These attributes can also be used to filter between spurious identifications and actual objects

of interest, e.g. using aspect ratio to remove small objects detected from the scene's background noise, or using the HSV histogram to filter false positive clusters of pixels that belong to the soccer field markings and not to a player.

After a proper tracking of an object across multiple sequential images, some other attributes can be estimated from the historical data, from the record of previous positions the speed and direction of the object can be estimated, which can be used to predict the object's next position.

Team Model

Similarly to an individual player, a group of players can also be identified by a set of attributes. In the case of soccer, the game is designed to have the players belonging to one of two teams, where the players of the same team wear uniforms of visual similarity (color, shape, patterns) with the exception of the goal keepers. Sometimes a spurious blob can pass a filter designed from considering individual objects, i.e. a player is expected to have some aspect ratio or a size within some margins, but spurious blobs can satisfy these conditions with unexpected shapes and produce false positives. Using the attributes derived from the common properties of a group of known players of the same team can be used as an additional filter where a blob is a player blob only if it fits the expected pattern of one of the two teams.

This useful team model, however, depends on the correct identification of a player, which can lead to a chicken and egg situation. In the case of knowing beforehand the initial position of the players, this piece of information can be used to create a trusty team model, but a bit of caution to keep in mind is that as the match advances and the conditions change (players' being replaced, the uniforms getting dirty changing the color distribution, abrupt scene light changes) this model can become obsolete, thus appropriate tools to keeping this model valid are required to make the most out of it.

(a) Yellow player with its hue channel histogram

(b) Blue player with its hue channel histogram

Figure 2.14: Comparing players from different teams

Player similarity

Having determined the model of properties that describes a player, its identity is spread to the next frame by comparing its set of attributes with the objects of the next frame that might represent the same player. If the objects seem closely related, the identity is spread, otherwise no connection is established between the objects.

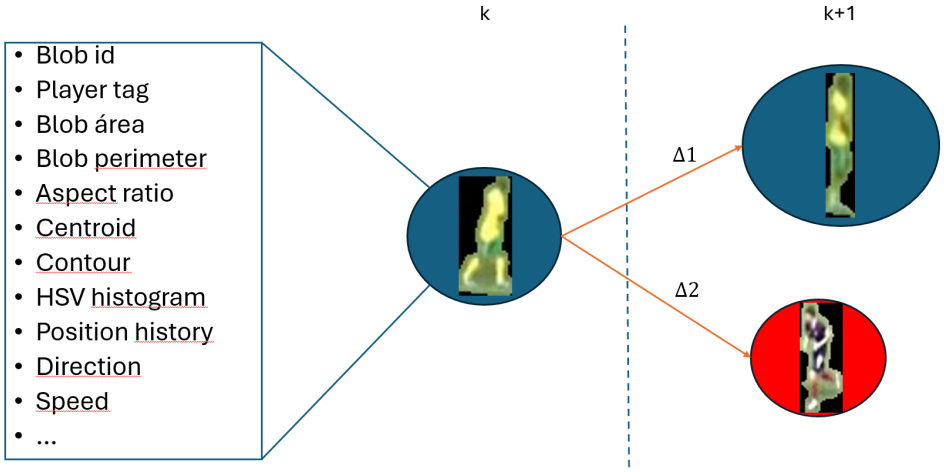


Figure 2.15: Using similarity between blob models to perform identity assignment across sequential frames

Each attribute is not treated equally, some have a greater impact in determining if the two objects represent the same player, as is the case of the position, taking into account the physical limitations of the players and the rate of the video recording (for example 30 FPS, two consecutive frames represent 1/30 of a second) it is impossible for one player to be positioned in one corner of the field in one frame and in the opposite corner in the next frame, the huge difference in the object's position rules

out the possibility of them representing the same player. In the other hand, other attributes are less influential when determining if the objects represent the same player, as shape or the contained area, as the players are expected to be humans running and kicking the ball with two legs. Not to say these traits are not useful or cannot be used to distinguish players, but differences between detected objects are expected to be low when considering this kind attributes.

Each attribute has its strengths and weaknesses when used for identifying objects (position helps filtering distanced players but becomes a lesser indicator when comparing objects close together), it becomes beneficial to use as many traits as available to cover a wider range of situations. Since some attributes can be used more generally than others, these are integrated into comparing object's similarity with a weighted value that can be adjusted to determine the influence of each attribute.

Thus, we can define the dissimilarity between two objects A and B as the weighted difference between the object attributes, where the lesser the value the more the similarity.

$$D(A, B) = \sum_{i=1}^{n} w_i \Delta(A_i, B_i) \tag{2.2}$$

Where $D(A, B)$ is the total dissimilarity between objects A and B, n is the number of attributes, w_i is the weight associated with the i-th attribute pair (A_i, B_i), and $\Delta(A_i, B_i)$ is the delta (difference) between the i-th attribute of object A and object B.

Next Position Prediction

When tracking projectiles or objects expected not to change much its trajectory [58], [77], a clear mathematical model that describes the object's trajectory is quite useful, whenever the optical tracking fails, the tracker can rely on the model to predict the object's position in subsequent frames until it is detected again by the tracker. The trajectory model can also be used to trace back in time the objects motion to estimate the trajectory in a time frame not present in the current data set. When tracking objects with more complex motions or movement freedom, these models can become complex to determine. In the case of human motion, in spite of having a general sense of where the person might go based on an objective (the players follow the ball from one side of the field to another), unknown factors can unexpectedly alter the behavior of the person, and thus its motion (the player tripped over, slowed down due to fatigue, got distracted by a flying bird, does not know where the ball is at and so it is not following it, it is positioning itself in advance of a future play).

PF comes handy in situations like these [32], [58], [65], where instead of predicting the players' next position, a cloud of points represent the space of possible next positions, the player is then predicted to be where most of these points cluster. This capability is not currently incorporated into ACE platform, and integrating it is a whole project on itself.

Assuming some constraints in the motion freedom of a soccer player, a simpler approach can be used. Considering the input video has a rate of 30 FPS, where the time change between two consecutive frames is 1/30 of a second, taking into account the distance from which the camera is set from the field to record the match, the pixel velocity can be calculated to describe the motion of players in the scene, which can be used to expose how much change are we to expect in between frames for a player in movement [33].

Keeping things simple, we can assume a rectilinear motion for immediate frames k and $k+1$, by knowing the player's direction and speed from the history of positions the immediate next position can be estimated. How well the prediction fits depends on how well we can estimate the player's speed and direction, which depends on the length of the previous positions history, too short a history makes the prediction too sensitive to new position information (direction may change on every new prediction) and too long a history will include outdated position information that may prevent the direction to be updated (the player moved to the right of the field for too long and then decides to turn the other side, the history of 'right direction' weight more in predicting the new direction than the new 'left direction').

Object Proximity

As stated before, distance between the position of objects is a useful metric that can be used to rule out relationships between objects that are too far apart. In the case of occlusion probability, the closeness between two objects can be used as an early indicator of an occlusion event. However, the distance between two objects is usually measured between the objects' centroids, but the occlusion can occur with any part of the region each object holds, a bigger object can occlude a smaller object even when the euclidean distance of the centroids is great. Figure 2.16 illustrates this problem by comparing two set of objects with the same euclidean distance from their centroids, top shows the bigger object partially occluding the smaller one, bottom shows no interaction between objects of same size.

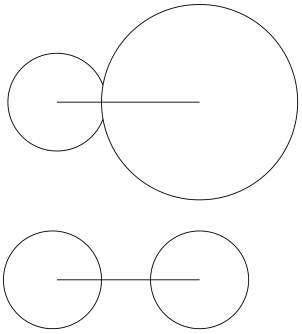


Figure 2.16: Problem with occlusion and euclidean distance as proximity metric

To address the previous scenario, the proximity between two objects is defined taking into account not only the (euclidean) distance between the objects, but also the size ratio between the objects (width and height ratios), including weighted values to provide flexibility to the calculation on how each attribute influences the proximity.

$$\text{proximity} = \text{distance} \times (w_{\text{distance}} + (w_{\text{sizeWidth}} \times \text{sizeRatioWidth} + w_{\text{sizeHeight}} \times \text{sizeRatioHeight})) \tag{2.3}$$

Where:

- distance is the Euclidean distance between the centers of the two objects, with (x_A, y_A) and (x_B, y_B) the positions of objects A and B, respectively.

$$\text{distance} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{2.4}$$

- sizeRatioWidth is the ratio of the widths of the two objects, with w_A and w_B the widths of objects A and B, respectively.

$$\text{sizeRatioWidth} = \frac{w_A}{w_B} \tag{2.5}$$

- sizeRatioHeight is the ratio of the heights of the two objects, with h_A and h_B the heights of objects A and B, respectively.

$$\text{sizeRatioHeight} = \frac{h_A}{h_B} \tag{2.6}$$

- w_{distance}, $w_{\text{sizeWidth}}$, and $w_{\text{sizeHeight}}$ are the weights assigned to each factor (distance, size ratio of width, and size ratio of height) to control their relative importance in the proximity calculation.

Object Area Overlap

In terms of occlusion events, it can be useful to know when an object might be take part in an occlusion event, one tool to determine it is by calculating the overlap between the bounding areas of two objects [47]. The overlap between two objects' bounding boxes is can be measured using the Intersection over Union (IoU), a standard metric in computer vision, and object detection and tracking.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.7}$$

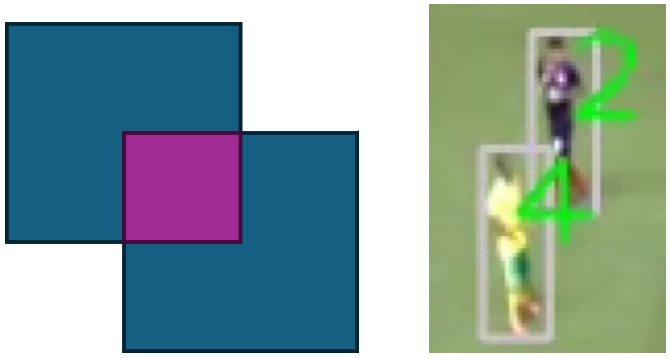
Where:

- The Area of Overlap is the region where the two bounding boxes intersect. We need to find the coordinates of the intersection rectangles (e.g., $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$) and calculate its area:

$$\text{Area of Overlap} = \max(0, x_{\max} - x_{\min}) \cdot \max(0, y_{\max} - y_{\min}) \tag{2.8}$$

- The Area of Union is the total area covered by the two bounding boxes.

$$\text{Area of Union} = \text{Area of Box A} + \text{Area of Box B} - \text{Area of Overlap} \tag{2.9}$$



(a) Graphical representation of area overlap

(b) Overlap of player's bounding box

Figure 2.17: Tracking object's area overlap

2.6 Multiple Object Tracking Metrics for Validation

The performance of the tracker can be evaluated by a set of metrics that expose different aspects of it by comparing a set of ground truth data with the set of results from the tracker. The ground truth

data is usually collected from the manual annotation of the players' positions on each image of a soccer match video with an annotation tool, a trained person marks the X and Y position of the player and sets a tag or label to mark the player's identity. The data of a single image can be used to evaluate the segmentation and identification algorithms, and the data of the sequential positions of the same tag across the video can be used to evaluate the performance of the tracking algorithms, i.e. how good the tracker is at detecting players and how good it is at tracking the players.

Petsas & Kaimakis worked with a static camera for multi-target tracking, aided with 3D data from Unity3D game engine simulation to generate the data sets [65]. The use of simulation can be considered for early testing of the algorithm, but there are conditions that won't match between real life and simulation (glitching in virtual world, motion that a real person can perform but a virtual 3D model can't). Generating data from a simulation can be useful in particular when reference data is scarce, it is not a secret that manually annotating object tracking ground truth data is an exhausting and not attractive task.

With the more recent advancements on AI, some tools have been released to aid in generating ground truth from real data, such is the case of CVAT [61], an AI powered tool. However, both the simulated soccer matches or the automated annotation tool require more effort and learning than the traditional manual annotation. Integrating these tools is, however, an improvement for further development.

Taking a blob as a cluster of pixels that is detected as an object of interest, the following basic metrics to evaluate the tracker can be defined as:

- False Positive (FP): Blobs that are not an object of interest - spurious blobs.
- True Positive (TP): Blobs that are an object of interest - player blobs.
- False Negatives (FN): Pixels or region of pixels not detected as a blob but that are part of an object of interest.
- True Negatives (TN): Pixels or a region of pixels not detected as a blob that are not part of an object of interest.

From the above mentioned metrics others can be calculated to evaluate the tracker on different aspects [48].

- False Positive Rate (FPR): Proportion of negative instances incorrectly classified as positive.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.10}$$

- False Alarm Rate (FAR): Rate of false positives relative to all negative instances, often equivalent to FPR.

$$\text{FAR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.11}$$

- Detection Rate (DR): Proportion of positive instances correctly classified as positive, also known as recall or sensitivity.

$$\text{DR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.12}$$

- False Negative Rate (FNR): Proportion of positive instances incorrectly classified as negative.

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{2.13}$$

- True Negative Rate (TNR): Proportion of negative instances correctly classified as negative, also referred to as specificity.

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.14}$$

- Accuracy: Proportion of correctly classified instances (both positive and negative) relative to all instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.15}$$

- Precision: Proportion of correctly classified positive instances out of all instances predicted as positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.16}$$

- Recall: Proportion of positive instances correctly classified as positive (same as detection rate).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.17}$$

- F-measure: Harmonic mean of precision and recall, balancing the two metrics.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.18}$$

In the particular case of multiple object tracking, two metrics are widely used to evaluate the tracker's performance. Although new metrics are proposed to better evaluate a tracker and analyze its performance on particular scenarios, MOTA and MOTP are the traditional choice.

- Weighted Multiple Object Tracking Accuracy (wMOTA): A metric that evaluates the spatial accuracy of a tracker's predictions for object locations by aggregating errors in detection and identification.

$$\text{wMOTA} = 1 - \frac{\sum_k (w_{\text{FP}} \cdot \text{FP}_k + w_{\text{FN}} \cdot \text{FN}_k + w_{\text{IDS}} \cdot \text{IDS}_k)}{\sum_k \text{GT}_k} \tag{2.19}$$

  Where: w_{FP}, w_{FN}, w_{IDS} are the weights for false positives, false negatives, and identity switches, respectively, FP_k, FN_k, IDS_k the number of false positives, false negatives, and identity switches at time k, respectively, and GT_k the total number of ground truth objects at time k. The summation is performed over all time steps (or frames) k.

- Multiple Object Tracking Precision (MOTP): A metric that measures the spatial precision of a tracking system by evaluating the average alignment between predicted positions and ground truth positions for correctly matched objects.

$$\text{MOTP} = \frac{\sum_{i,k} d_{i,k}}{\sum_t c_k} \tag{2.20}$$

  Where $d_{i,t}$ is the distance (e.g., Euclidean or IoU-based) between the predicted position and the ground truth for object i at time k, and c_k is the number of matches (correct associations) at time k.

New interesting benchmarks are being discussed that are worth to pay attention to, as the results of these new metrics can expose weaknesses and strengths of the algorithms that might remain hidden when evaluating with a single general metric [12], [56]. Some of the limitations of MOTA metrics are [44], [56], [71]:

1. **Aggregation of Errors:** MOTA combines different types of errors into one score, which can obscure the relative significance of each error (e.g., false positives vs. identity switches).

2. **Bias Toward Crowded Scenarios:** In datasets with many ground truth objects, MOTA may not reflect the severity of errors in sparse environments, since errors are divided by the number of objects.

3. **Ignores Localization Accuracy:** MOTA does not consider how accurately the tracker localizes objects, focusing only on detection and identity consistency.

4. **Sensitivity to Matching Threshold:** MOTA's score is sensitive to the threshold used for matching (e.g., IoU), which can vary across benchmarks and implementations.

5. **Penalizes Small ID Switches Heavily:** MOTA treats all identity switches equally, regardless of their duration or frequency, which may not always be a fair representation of tracking quality.

6. **Lack of Temporal Behavior Insight:** MOTA does not distinguish between occasional and consistent tracking errors, making it hard to evaluate long-term tracking performance.

7. **Negative MOTA Scores:** MOTA can result in negative scores if the total errors exceed the number of ground truth objects, which can be counterintuitive.

8. **Poor Differentiation Between Trackers:** In simpler tracking scenarios with fewer objects, MOTA scores for different trackers can be similar, making it hard to distinguish performance.

9. **Inability to Handle Partial Matches:** MOTA does not account for partial matches (e.g., partially occluded objects), which can lead to less accurate evaluations in challenging scenarios.

10. **Application-Specific Limitations:** MOTA assumes equal importance for all error types across tasks, which may not align with the priorities of specific applications, such as autonomous driving or surveillance.

To address these inconveniences, new metrics and benchmarks are in continuous development, however, these are not explored as part of the current project, but a variation of MOTA is utilized to evaluate the tracker, this weighted variation assigns different weights to false positives (FP), false negatives (FN), and identity switches (IDS), having these weights allows us to evaluate the tracker on different aspects, i.e. we can focus on occlusion events by maximizing identity switches errors and removing false positives from scope, considering false positives come from segmentation and detection stages.

$$\text{wMOTA} = 1 - \frac{\sum_k (w_{\text{FP}} \cdot \text{FP}_k + w_{\text{FN}} \cdot \text{FN}_k + w_{\text{IDS}} \cdot \text{IDS}_k)}{\sum_k \text{GT}_k} \tag{2.21}$$

Where: w_{FP}, w_{FN}, w_{IDS} are the weights for false positives, false negatives, and identity switches, respectively, FP_k, FN_k, IDS_k the number of false positives, false negatives, and identity switches at time k, respectively, and GT_k the total number of ground truth objects at time k. The summation is performed over all time steps (frames) k.

Chapter 3

Methodology

The following tasks and considerations are required to achieve completeness of the projects' objectives, separated into three phases: Structure of theoretical background and state of the art, Implementation of occlusion handling into ACE platform, and Validation of tracking tool.

1. Structure of theoretical background and state of the art

    - Review professional literature related to multiple object tracking, sports analysis and state of the art methods and algorithms on the subject. The literature to be reviewed is maintained in trusted repositories, as IEEE Explore®, ACM®, ScienceDirect® and Springer®.
    - Structure reviewed literature according to distinct approaches on multiple object tracking and occlusion detection and handling.
    - The algorithm to be implemented is an improvement on the current development of the automatic players' tracking platform ACE, therefore, this platform functionality and structure will be studied in order to integrate multiple hypothesis tracking.
    - Define occlusion: occlusion detection, occlusion identification, occlusion handling.
    - Study how occlusion occurs and its effects on optical tracking of soccer players.

2. Implementation of occlusion handling into ACE platform

    - The platform to be used is composed of already existing libraries written in C++, from the platform ACE and other trusted open-source developers as OpenCV, as well as libraries to be developed according to the needs of the project.
    - Multiple Hypothesis Tracking algorithm implementation is to be studied from professional literature that depicts its implementation and special considerations, as well as from different already implemented free access libraries, to develop a custom library that covers ACE's platform needs.

- Alternatives to Multiple Hypothesis Tracking (MHT) are to be considered in order to increase the tracker's performance metrics.
- ACE tracking platform code is to be maintained in a private repository hosted by the Laboratory of Pattern Recognition and Intelligent Systems PRIS-Lab.

3. Validation of tracking tool

- The validation will be performed using a manually annotated ground truth database generated through an Annotation Tool.
- An Annotation Campaign will be organized and executed to facilitate the generation of ground truth data by the collective work of volunteers.
- The ground truth data includes the 2D position of soccer players from a soccer match video, the tool will include a means to identify occlusion events, thus allowing to generate an occlusion ground truth data set for validation.
- A clip of a soccer match video will be selected based on the range of interactions within it. As the main thread of this project is to evaluate occlusion events, the tracker will be evaluated using extracts of a soccer match where occlusion are clearly present.
- The principal metric to be used to evaluate the tracker performance is the Multiple Object Tracking Accuracy (MOTA). Other metrics are included to provide a more complete analysis of the tracker's results.
- The tracker's results are to be compared against the tracking results of the previous release of ACE tracking platform using the same input video and ground truth data set.

Chapter 4

Implementation

This chapter covers the implementation of Multiple Hypothesis Tracking (MHT) and Multiple Multipartite Graphs (MMPG), as well as complementary logic added to the tracker to increase its performance. Sections 4.3, 4.4, 4.2 and 4.6 cover algorithms and processes designed to improve the tracker's tracking capabilities, whereas section 4.5 depicts an improvement in the segmentation and detection phase, designed to provide a cleaner input data set to the tracking phase.

ACE platform is written in C++11 coding language, it uses third-party libraries for specific utilities (OpenCV image manipulation, yaml-cpp to process input/output files, FFMPG for video and image processing, ALGLIB for numerical analysis and data processing), but its main framework is custom code developed for the particular use of the platform, the result of many iterations in collaboration with students and researchers. The platform is designed to run on Linux environment, on Windows 10 and above we can use Windows Subsystem for Linux (WSL) or a virtual machine. Code development is done in JetBrains CLion IDE using a student license [37].

4.1 Multipartite Graph Tracking

ACE tracking platform is based on the implementation of a multipartite graph tracker where each graph contains all the extracted information of interest from an image frame, sequential graphs represent sequential frames of the input video. Each node inside a frame represents an individual blob and its attributes that distinguish it as a unique object, and that can be used to filter spurious blobs from player blobs, where a player blob represents a soccer player and a spurious blob represents a group of pixels mistakenly detected as a soccer player.

The tracker flow can be summarized in the following steps:

1. Preparation stage
    - The configuration parameters are interpreted from the configuration file.
    - Initial state of the players is stored from user input.

- The input video is processed. General information about it is extracted and stored as reference.
- The required frames are extracted to start the batch processing.

2. Segmentation and detection stage

- Each frame is processed using different filters to extract the background and segment the players. In particular, HSV channel data is extracted to extract the green field background.
- Blobs are extracted and assigned attributes, including identification.
- Players' initial state is used to identify players in the first frame. Euclidean distance to closest blob is used to determine this initial assignment. Some player blobs can be distinguished from this step.
- Extracted blobs are processed to filter out spurious blobs by aspect ratio and size.
- Remaining blobs are considered player blobs and comprise the nodes or vertices of the frame graph. The Multipartite Graph (MPG) is built from each new graph.

3. Tracking stage

- The MPG is processed to generate edges between the vertices of graph k and the vertices of graph $k+1$.
- Nodes from graph k are compared against nodes of graph $k+1$. A similarity score is estimated using the node's (player) attributes. Only the best similarity is preserved to create an edge between nodes.
- All blobs (nodes) are re-evaluated to filter spurious blobs. Assignment history is used to aid the filtering.

4. Results preparation stage

- The resulting MPG is used to generate the batch tracking results, which are stored in an external file. This is done so as to limit the memory consumption and re-utilize memory resources.
- Visual markings are drawn to the respective frames and saved to an external video file, which is updated with every processed batch.

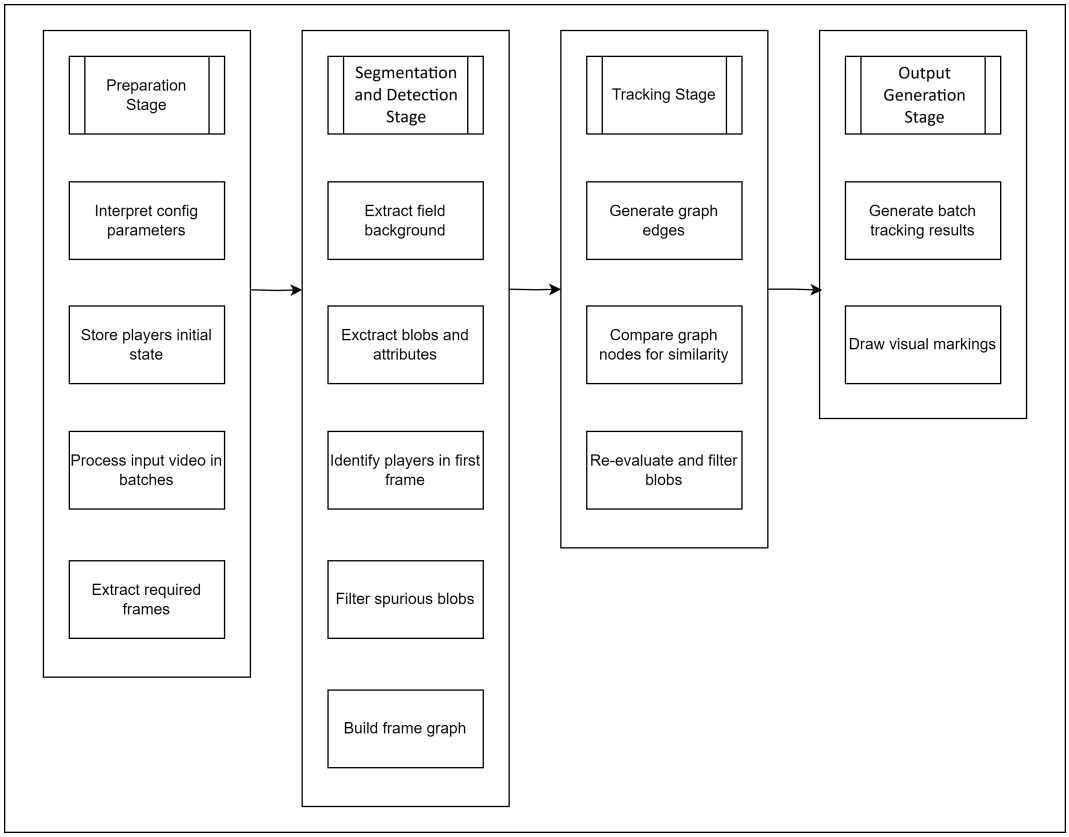
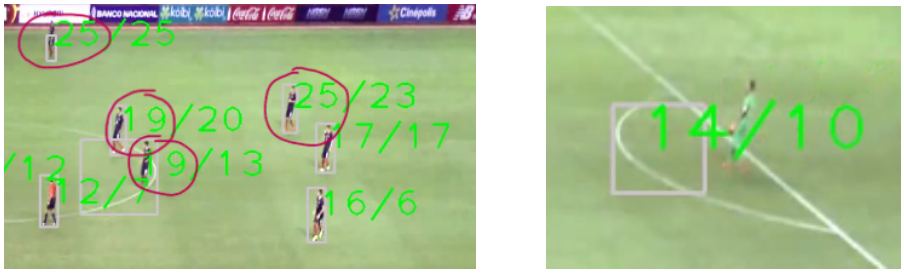


Figure 4.1: Flow diagram of ACE multipartite graph tracker

The MPG infrastructure is robust enough to recover some of the missing identities and detect blobs merging, specially with the use of an assignment history or *memoirs*, useful when a player is lost in a couple of frames and then detected in the following ones.

However, this approach is not without faults. The edges are assigned based on the best similitude of the isolated graphs that are being compared, resulting in the possibility of assigning a local minimum that satisfies the mathematical comparison between nodes, but that in reality is a misassignment from the lack of information of the global scope. More particularly, noise (spurious blobs) and missing identities do not stop the algorithm from assigning the best local match, e.g. if player P1 from graph k is missing in graph $k+1$, the algorithm will try to match it with the available nodes in that graph, even though the actual node the represents P1 is not present. Also, as the assignment considers only the best edge for each individual node, it is possible to link more than one node from graph k to the same node of graph $k+1$, and vice versa, which leads to duplicate identities that are spread to the subsequent graphs evaluation. These errors satisfy the mathematical constraints, but are not correct from a semantic perspective.



(a) Duplicate identities in ACE tracking: two players with ID 25 and two players with ID 19 in the same frame

(b) Spurious blob detected as a player instead of the actual player

Figure 4.2: Examples of errors in ACE tracker identity assignment

4.2 Duplicate solver

Each blob structure holds two identities: the blob identity and the player tag. The blob identity ensures the object's uniqueness in the same graph, this does not prevent two blobs of holding the same attributes, but is needed to process the graphs. The player tag relates a blob object with a player, this tag is the one used to track the player's state throughout the video, the final trajectory of a player is determined by the position of each blob identified as that same player, ordered by the graph sequence within an MPG. Ideally, only one blob can hold the same player tag within the same graph (players are unique), however, this is not the case as the conditions of the scenes are not ideal, the presence of spurious blobs, false negatives, blobs with similar attributes (look-alike players) and occlusion set the conditions in which solving the MPG is satisfied with duplicated identities, i.e. two blobs different blob identity share the same player tag.

To address this not ideal result, a constraint in the uniqueness of player tags within the same graph is integrated to the MPG assignment after edges resolution. When two player blobs share the same player tag in graph k, the tracker will get the player blob from graph *k-1* with the duplicated player tag we want to fix and calculate again the similarity between the player blob from graph *k-1* and player blobs with duplicated tag from graph k, the identity is kept in the player blob with the best similarity. Next, the tracker will get the closest player blobs from graph *k-1* (calculates the proximity based on 2.5 and a predefined threshold) and the remaining player blobs from graph k whose player tag was stripped after solving the duplicate conflict, then, tracker calculates once the similarity between the objects in graph *k-1* and the player blobs from graph k without player tag. If the remaining objects from graph k are similar to an object from *k-1* whose identity was not spread to the next frame, its

identity is assigned to that object in graph k, otherwise, it is assumed that the object in graph k that got its player tag stripped holds a new identity. This scenario is possible when players are replaced or ejected from the game. But it can also mean that the tracker failed in detecting a player (missing identity) and failed to associate its identity to an object when the player was detected again, the same holds true to the duplicate identity scenario.

(a) Duplicate identities in ACE tracking **(b)** Fixed duplicate identities

Figure 4.3: Examples of duplicate identities error in ACE tracker fixed by duplicate solver

In an ideal scenario, solving duplicates can help recover missing identities and cover the players' replacement event. However, as the tracker has to deal with noisy detections, missing identities can happen more than often, leading the tracker into considering all objects with not fixed identities to be treated as new objects in scene after solving duplicates conflicts. One improvement is to look into 'the past' more than one frame, but special considerations are to be taken in terms of proximity, as the more frames backwards the more spatially distant the objects can be.

4.3 MHT

As explained in [6], MHT can traditionally be implemented in two different ways: Hypothesis Oriented and Track Oriented. Find below a brief comparison between the two methods, not to be considered as a definitive nor comprehensive description, but as an easy to digest summary.

Aspect	**Hypothesis-Oriented MHT (HOMHT)**	**Track-Oriented MHT (Track-Oriented MHT)**
Core Concept	Focuses on hypotheses about *data associations* at each frame.	Focuses on hypotheses about *entire tracks* over multiple frames.
Hypotheses Representation	Hypotheses represent all possible *detection-to-track associations* for a single frame.	Hypotheses represent *object trajectories* (sequences of states) over time.
Decision Granularity	Makes decisions frame-by-frame.	Makes decisions for full tracks (spanning multiple frames).
Deferred Decision-Making	Delays decisions about associations until sufficient evidence across a few frames is available.	Delays decisions for longer periods, evaluating track consistency over multiple frames.
Ambiguity Handling	Resolves ambiguities by maintaining multiple hypotheses for each frame.	Resolves ambiguities by maintaining multiple possible tracks over time.
Hypothesis Tree Growth	Branching occurs with each new frame based on new detection-to-track associations.	Tree grows based on possible continuations of object trajectories.
Scalability	More scalable in highly dynamic scenes since decisions are localized to each frame.	Computationally expensive for large numbers of objects due to longer hypothesis maintenance.
Pruning	Prunes hypotheses for each frame based on their likelihoods.	Prunes entire track hypotheses to manage exponential growth of possibilities.
Complexity	Lower complexity, as hypotheses are limited to a single frame's associations.	Higher complexity, as the tree grows to represent entire tracks across multiple frames.
Accuracy	Handles short-term ambiguities effectively.	Better for long-term tracking, especially in occlusions or missed detections.
Decision Scope	Decisions are made for individual associations at each frame.	Decisions are made for entire object tracks over time.

Table 4.1: Comparison of Hypothesis-Oriented MHT and Track-Oriented MHT [27], [68], [70], [84]

After evaluating which approach to implement, the following are the main considerations into selecting Track Oriented Multiple Hypothesis Tracking (TOMHT) over HOMTH to integrate MHT into ACE platform:

- The tracker processes the video by batches of images, as MPGs are created for the entire batch, considering an MPG solution as a single hypothesis more closely relates to TOMHT decision granularity in spans of multiple frames.
- The desired tracking output is trajectory of each individual player (the history of positions) defined by the set of player blobs of each graph in the MPG with the same player identity. This output resembles a single track per player from a TOMHT structure.
- In the case of the Hypothesis Oriented Multiple Hypothesis Tracking (HOMHT) and ACE platform, a new hypothesis is created for each possible set of assignments in a single frame, including the assignment of new object or dead object. Assuming all possibilities in the space of fixed number of players in a whole match, hypothesis grow at a rate of , considering the batch processing we end up with (not considering pruning). In the case of TOMHT, a new track is added only on new objects detection, assuming a space of fixed number of players, we end up with a fixed number of tracks maintained for the whole match, each track with as many levels as frames in the batch.
- Holding a single track per player can help to cover the duplicated identities issue from MPG described in 4.2.
- Implementing HOMHT in ACE tracker would require greater changes in the framework than implementing TOMHT, MPG structure can be reused to integrate TOMHT.

As already mentioned, in a TOMHT a single track hypothesis is created for a unique object of interest, a new branch is created as new identity assignments are done for the object of interest; considering a top-down hierarchy, the track hypothesis is comprised of assignment levels, where each level represents the possible detected objects assignment in a particular frame, where subsequent levels represent sequential frames.

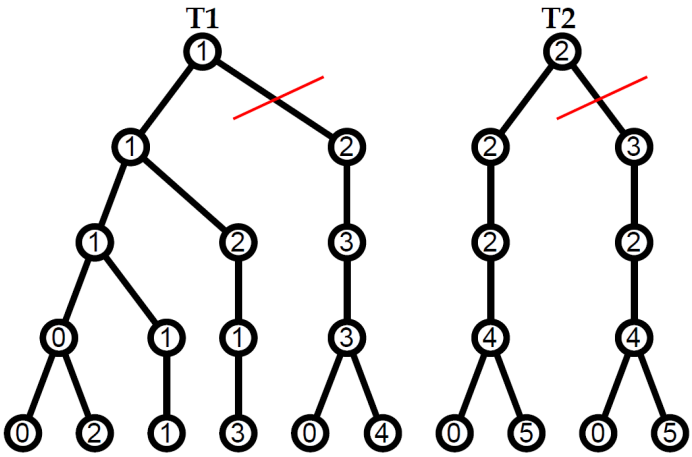


Figure 4.4: TOMHT: a single track tree represents a single detection, the branches are the assignment hypothesis. Vertical levels represent frames, nodes in the same level are assignment hypothesis of the tracked object in the respective frame. [6]

TrackHypothesis

Attributes:

- unsigned root_level
- unsigned int last_level
- unsigned int first_tracked_level
- unsigned int number_untracked_levels
- std::map<unsigned int, std::map<int, double>> levels
- std::string track_name
- int track_id

Methods:

+ InsertTrackNode(unsigned int level, int blob_id, double value)
+ PrintTrack()
+ GetLevelHypothesisByWeight(unsigned int level) : std::queue<std::pair<double, int>>
+ PruneByAvg()
+ RemoveTrackNode(unsigned int level, int blob_id)
+ GetBestLevelBlobId(unsigned int level) : std::pair<int, double>

Figure 4.5: TrackHypothesis UML

The hypothesis are managed as follows: the track determines the player identity, the level identifies the frame of the assignment, each node/branch is an assignment hypothesis where we relate the player identity (track) with the detected player blob (node), i.e. what blobs in the frame might represent the tracked player. To make use of the MPG data structure, the player blob data structure is updated so that each node holds its next frame possible assignments or hypothesis space, allowing TOMHT and MMPG to coexist. Once all hypothesis are created for the current batch, the required tracks are generated with the hypothesis from the nodes and then the assignments are resolved.

Figure 4.6: PlayerBlob updated structure to support hypothesis

Assignment is determined by similarity to the previous levels, the root of the track represents the player's initial state (trusty known state at the beginning of the match from user annotated input). A similarity threshold is used to keep only certain possible assignments, preventing uncontrolled growth in track branching for each level, the final identity assignment is solved when the track has all the necessary levels (all frames in the batch processed), the best branching path is selected, taking into account that player blobs are unique and so are the tracks (one-to-one relationship between tracks and players), the same player blob cannot exist in more than one track in the same level, this constraint helps in preventing duplication of identities. This approach differs from the MPG one in that it is acceptable to not have a match between nodes in a particular frame, whereas in MPG all nodes are expected to have a match, forcefully matching can create duplicate identities and cover miss-detections.

A couple of problems worth mentioning that might arise with TOMHT are that if the tracker struggles in detecting objects continuously (intermittent players), many of these detections can be interpreted as new objects that will create new tracks, this is particularly problematic if there are many recurrent spurious blobs. Another problem is that the assignment probability is based of similarity, too similar objects can cause identity swaps.

4.4 MMPG

Taking advantage of the non-minimized MPG, the multiple hypothesis can be held in a single non-minimized MPG as discussed in 2.4. When creating edges between nodes from graph k and nodes from graph $k+1$, multiple edges per node are allowed instead of only keeping the best, the hypothesis growth is controlled by a threshold defined by how far a player can move in between frames (this same threshold exists in the MPG) so that an edge is created only between nodes in a close space of a player's next possible positions. The MMPG is minimized using all the created edges (hypothesis).

Since this approach is based on MPG, some issues are inherited. Without a constraint in unique identity assignment, duplicate identities are possible, as this scheme is not precisely MHT, where each hypothesis is constrained to enforce unique assignments, but an adaptation of the idea on holding a set of possible solutions and delaying assignment resolution until after the whole hypothesis space is created. Another issue that comes to play is the stitching between MPGs of consecutive batches, an occlusion event might split into two batches (as discussed in 4.1). To address the duplication issue, the tracking could be aided by 4.2; to address the batching split we can add flexibility to the tracker so that batches are not of fixed size, which is discussed in C.

4.5 Filtering spurious

One big issue uncovered during the MHT and MMPG implementation is the high impact of the spurious blobs in the tracking results. Even after removing spurious blobs by size, ratio, HSV filter and idleness [92], an important number of spurious blobs remain, which once processed by the any of the aforementioned tracking approaches will mess up the identity assignments. After the spurious filtering, these spurious blobs will be considered as player blobs, and therefore are expected to be assigned some player identity. In the best of cases, these blobs are assigned a new player identity and can later be caught by the idleness filtering using memoirs (history buffer of existent blobs); in the worst case, the blobs will assume an existent player identity, causing duplicates, identity switches and covering up missing identities.

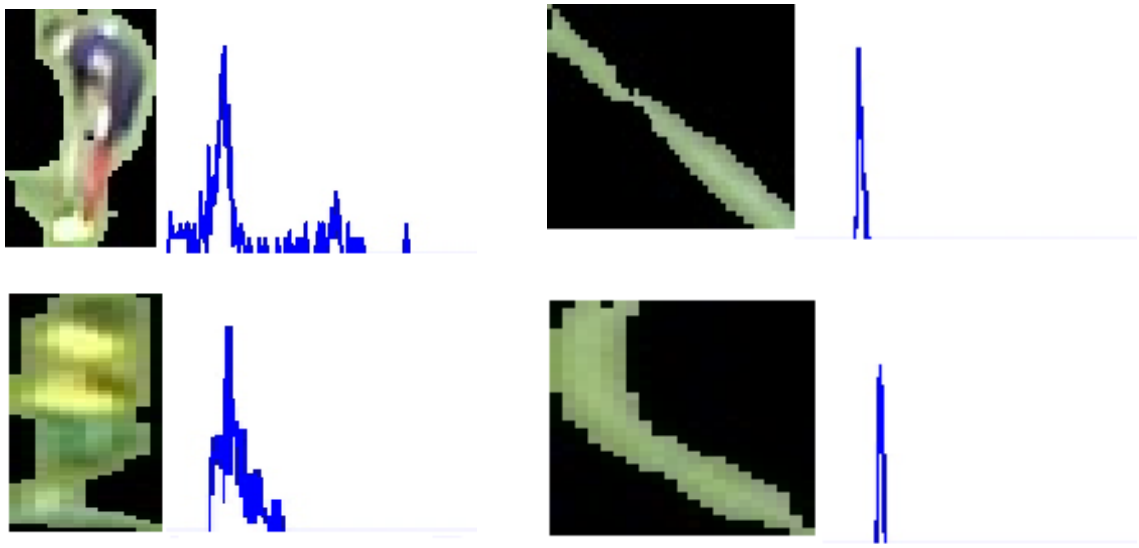
Template matching is widely used in optical object tracking applications [46], [82], its main idea is to have a template that represents the expected appearance of the object of interest. Based on this core concept, in 2.5 we discussed the idea of creating a model of attributes that represents a group of similar objects; any outlier blob is to be considered a spurious blob assuming all players from the same

team share similar optical attributes.

To create this team model, we start by identifying the actual players to ensure we are not including spurious blobs into this model. ACE platform can receive an input set of positions that determine the initial state of the players, usually reserved to the start of the match, from these positions the platform can determine initial assignments to the detected blobs, any not initially assigned blob is not considered a player blob, but also not yet considered a spurious blob, as the tracker might fail in assigning all of the expected initial identities (it can fail in detecting a player from the very start).

After identifying the player blobs, they are assigned to one team or the other by the position on the field, left team and right team. The hue channel histogram of each player is generated from the pixels within the contour area, each histogram is stored as part of the team model attributes (see 2.5 figure 2.14). The Bhattacharyya distance [21] of each histogram against all other histograms in the same team group is calculated and averaged (one avg per player in the team), only the greatest average is stored as a team model property, to be used a threshold on the furthest two histograms can differ for a blob to be considered part of the team.

Once both team models are complete, the blobs' hue histograms are compared against each of the hue histograms from the team model using the Bhattacharyya distance and averaged. The average Bhattacharyya distance of the blob under study is then compared with each team models max average, which serves as the threshold, a blob is considered a player blob only if its average Bhattacharyya distance is greater than both team models max average Bhattacharyya distances. Only by adding this new filter, the false positives in the test video dropped by 26.97%.

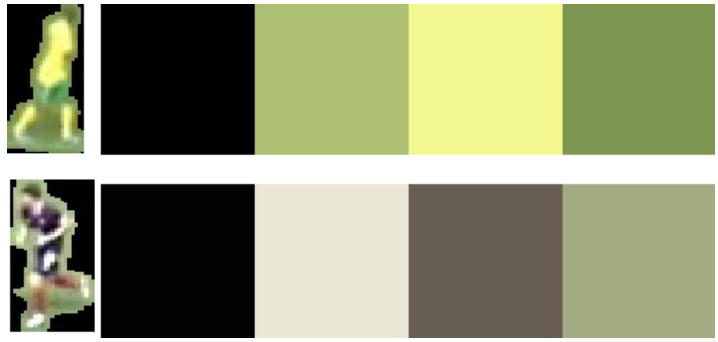


(a) Player blobs **(b)** Spurious blobs

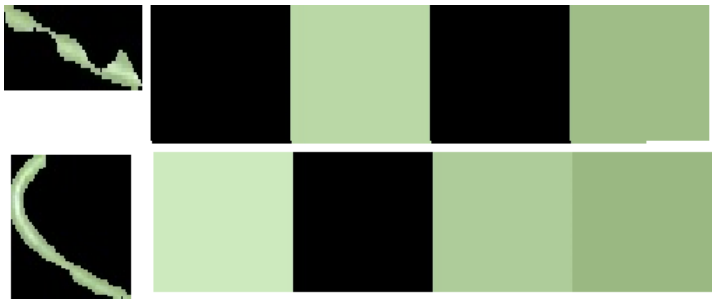
Figure 4.7: Spurious blobs vs Player blobs hue channel histogram filtering

However, since this method relies on the hue histogram distribution within a group, it is still possible for a spurious blob to fit in, specially when the players' histogram is closely similar to the field background, e.g. players wearing uniforms in the green hue spectrum.

To make the filtering more robust, an additional step that compares the predominant colors is added. When the predominant colors of a player blob are extracted, we can notice a range of colors even in players with uniforms that resemble the field background, but when we look at spurious blobs, due to the consistency of the colors within the contour (mostly field, grass, maybe some field markings that are white), green remains the predominant color. Moreover, when we take not only the blob contour but the binding box, the color black becomes the predominant color, and as we increase the numbers of predominant colors to extract, the more times the color black is listed in the spurious blobs, whereas other colors are extracted from a player blob before we reach the point of black repetition. Taking advantage of this pattern, an arbitrary number of colors is extracted from each blob, and any blob with more than an expected number of predominant colors being black is deemed as a spurious blob. Only this addition reduced the false positives in the test vide by 23.6%.



(a) Player blobs



(b) Spurious blobs

Figure 4.8: Predominant color swatches: Spurious blobs vs Player blobs. Notice in figure 5.11c that some spurious blobs can still pass the black predominant color filter.

(a) Blobs after segmentation in unchanged ACE platform

(b) Blobs after filtering spurious using **hue filter** in updated ACE platform

Figure 4.9: Example of effectiveness of the **hue filter** in removing spurious blobs

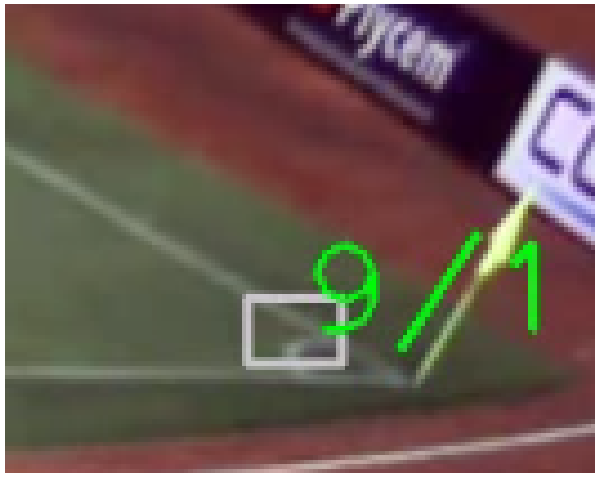
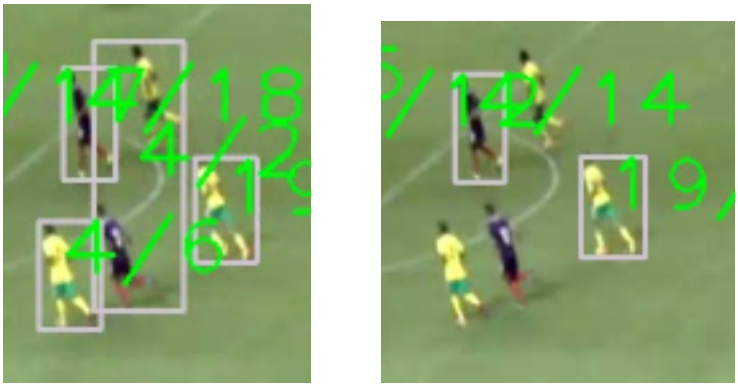


Figure 4.10: ACE Platform unchanged version wrongfully detecting the corner markings on the field as an object of interest

4.6 Handling Occlusion

One problem pending to address with our MHT and MMPG is the case of missing identities. Even though MHT is designed for situations in which noise or occlusion affect the correct identity assignment in Multiple Object Tracking (MOT), it is an algorithm that depends on the availability of data (nothing unexpected), in the case of an occlusion event, it is expected that the occluded object appears again in scene after the occlusion event concludes, this reappearance can be used to fill in the gaps caused by an occlusion event. However, in many cases, and more than often with ACE platform, the occluded object is not detected again after an occlusion, which can be rooted to the segmentation stage in some cases. It can also happen that the object leaves the first occlusion to immediately enter another occlusion, rendering the algorithm useless if there is no data to fill in the gaps. In figure 4.11, even though not a real occlusion event but an error in the segmentation of the players, notice that in frame *k* the players are identified but in frame *k+1* three players are missing, occlusion handling nor a multiple hypothesis approach can resolve this situation as there is no detection in frame *k+1*, there is no identity assignment to fix/resolve as there are no detected objects to identify or to fix the identities.



(a) Players in false occlusion event frame *k*

(b) Players in false occlusion event frame *k+1*

Figure 4.11: Spurious blobs vs Player blobs hue channel histogram filtering

From [39], [42], [100], [101] we can notice a pattern on occlusion solving with MHT in which MHT is used in conjunction with other prediction algorithms that can help recovering missing identities. This trend is put to test following the core concept from [60], an occlusion alarm is added to determine the start of an occlusion event by the objects' bounding box overlapped area, once an object is occluded, its last know state is used to fill in the gaps during an occlusion event. In the case of ACE platform, this new object created from the previous state of a missing object is named a *Ghost Player*, since the

attributes that determine it as a player blob are not extracted directly from the image but instead from a player that is no longer with us.

Ghost Player

A *Ghost Player* consists of a player blob in graph *k* that is created based of the properties of a player blob from graph *k-1* with a player identity that is missing in graph *k*. In terms of code, the player model is the same and are distinguished by a boolean flag *ghost_blob*. The main goal of a *Ghost Player* is to fill in the gaps of a missing player blob, so that it can be used during the assignment resolution.

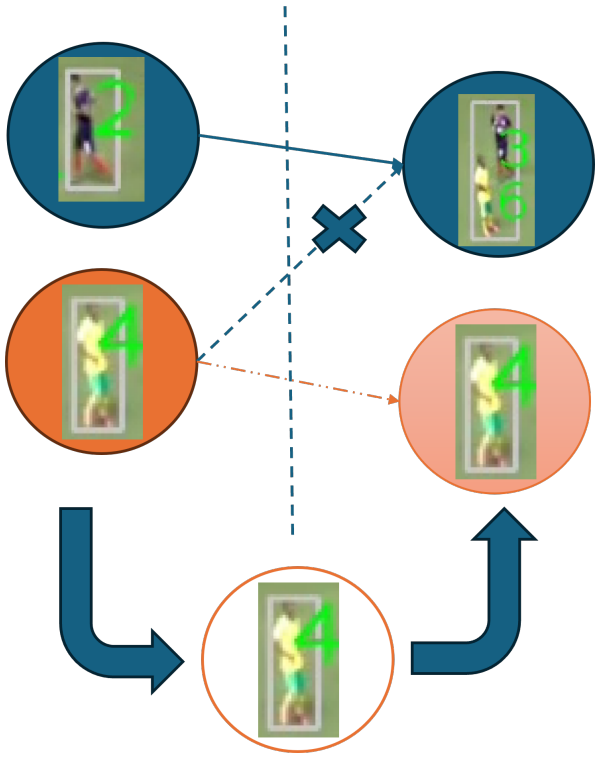
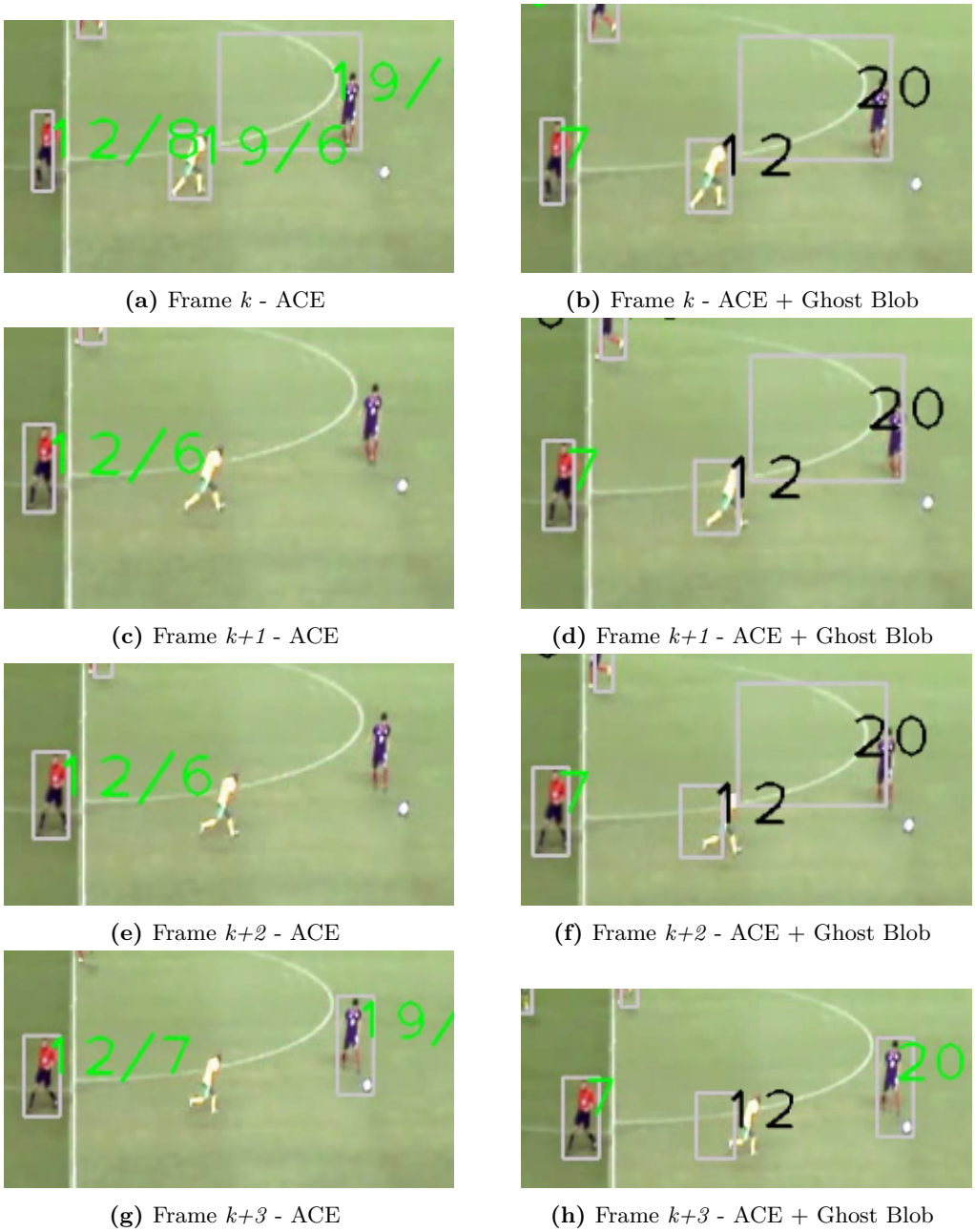


Figure 4.12: Ghost Player interaction in the graph tracking representation

A *Ghost Player* inherits the attributes of the missing player, however, as it is to be used as the actual missing player, its attributes are updated accordingly. From the history of positions, speed and direction, the new position is estimated as discussed in 2.5, the history of positions, speed and direction are also updated based of the new position. If the occlusion is long enough, we might end up in a situation in which the motion prediction is based solely on previous predictions, reducing the confidence of the prediction on every iteration and making the *Ghost Player* obsolete and a spurious

to the tracker.



(a) Frame *k* - ACE

(b) Frame *k* - ACE + Ghost Blob

(c) Frame *k+1* - ACE

(d) Frame *k+1* - ACE + Ghost Blob

(e) Frame *k+2* - ACE

(f) Frame *k+2* - ACE + Ghost Blob

(g) Frame *k+3* - ACE

(h) Frame *k+3* - ACE + Ghost Blob

Figure 4.13: Ghost Blob wit drifting next position prediction

To exorcise the *Ghost Blob* in time so it does not become a spurious, a check is required after its appearance to determine its value in filling the gaps. One way to do so is by extracting new visual properties from the updated bounding box, if the *Ghost Blob* encloses a player, then the prediction remains valid, but if the blob no longer encloses a player, the prediction has become inaccurate and

the blob obsolete. Using the team model as described in 4.5 we can determine if the new enclosed pixels belong to a player or not. It was considered to use the missing player's previous state as the point of comparison, but during an occlusion event the missing player is being occluded and thus the new bounding box might not include the missing player, rendering the comparison useless.

Ghost Blobs are quite situational and its use is in *hoping* we can predict the missing player good enough until the tracker is able to detect it again.

Occlusion Alarm

The occlusion alarm is a flag set to each player blob that is soon to be part of an occlusion event. Occlusion event occurs when two or more objects get too close together from the camera point of view (see 2.2), we can use this information to try to predict when two objects will face an occlusion event. [60] use the area overlap percentage of the objects' bounding box, following this idea, the overlap percentage of the player blobs' bounding boxes are calculated, if the percentage crosses a threshold then the alarm is triggered. Moreover, in ACE platform the occlusion alarm is also triggered by the euclidean distance and the proximity being too short (see 2.5 and 2.5).

When a flagged player blob in graph k is missing in graph $k+1$, the assumption is that the player enter an occlusion event which occluded it and thus the missing identity. From this assumption, a *Ghost Player* is created for the remaining of the occlusion event, or when the *Ghost Player* becomes obsolete. If the player is detected again, the *Ghost Player* is removed from the scene.

Player Blob Merge Splitter

ACE tracking platform has the capability of identifying some merging and split events based on the MPG edges associations, blob area and template matching [64] (to find player blob to recover inside the merged blob area). The tracker utilizes the memoirs (buffer of blobs history) to recover identities from a merge event. However, not all merge events as resolved, and some remain only with a tag identifying a merged blob as a player blob resulting from a merge event, including a list of the player identities that got merged, i.e. identities from graph $k-1$ are included as attribute of merged blob in graph k. Unfortunately, location of the object within the merged blob is not stored, as the retained information is from the player blobs from previous graph.

To expand this capability, the identified not resolved merged blobs' identities are split into individual player blobs based on the immediate previous state (instead of relying on memoirs). The player

identity from each blob involved in the merge in graph k is searched in graph *k-1*, once verified its previous existence, the player blob's next position is estimated and inserted into the graph k (see 2.5, 4.6). If all of the blobs from the merged blobs list are resolved, then the merged blob is removed from graph k (fully resolved), otherwise, it remains as some player blobs remain merged.

(a) Frame *k* - ACE

(b) Frame *k* - ACE + Merge Splitter

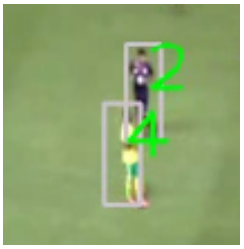
(c) Frame *k+1* - ACE

(d) Frame *k+1* - ACE + Merge Splitter

(e) Frame *k+2* - ACE

(f) Frame *k+2* - ACE + Merge Splitter

(g) Frame *k+3* - ACE



(h) Frame *k+3* - ACE + Merge Splitter

Figure 4.14: Splitting merged blobs

Chapter 5

Results and Discussion

This chapter exposes the most relevant results of this project, as well as interesting findings, and the strategy followed to generate them.

5.1 Test Strategy

ACE tracking platform has a complex structure with many configurable parts, the control parameters are defined in a configuration file that the tracker consumes prior to start processing the input video. To make the results comparable, a fixed set of configuration parameters are selected and maintained throughout the testing, only configuration parameters that directly affect the aspects under test are modified between executions, like enabling or disabling Multiple Hypothesis Tracking (MHT). The main point of comparison is the previous version of ACE tracking platform, provided with a working environment in a virtual machine image, this version of the tracker is used with the same applicable configuration parameters, processing the same input videos for the same range of frames and consuming the same players' initial positions file. The initial positions of the players are manually annotated in the first video frame to process utilizing an annotation tool.

ACE tracking platform will process the same input video for the same specific frames under different combination of enabled features so as to have visibility of the effect of each, helpful into finding the best combination of features and to identify those that have low benefit in the tracking results. The main features to test are as follows (identifier in parenthesis):

- ACE without modifications (original)
- Multiple Hypothesis Tracking (mht)
- Multiple Multipartite Graphs (mmpg)
- Hue filter (hue)
- Merge Split (ms)

- Occlusion Alarm (oa)
- Duplicate Solver (dp/dp_rm)

The results from the previous version of ACE are tagged as 'original' and the refactored ACE tracker without any new feature enabled as 'rework'.

The metrics are generated using the validation tool developed during [92]. The ground truth data set is manually generated using the annotation tool described in 5.1. Both, player's position and occlusion events data is incorporated to evaluate the performance of the tracker.

Occlusion events have been picked based on visual selection during the ground truth annotation process, the Occlusion Event Complexity Score (OECS) is calculated to classify the events, scenes with multiple occlusion events are classified by the highest OECS, metrics evaluation of individual occlusion events is not performed, the tracker is evaluated for the whole set of results in a scene, i.e. if the tracker solves a Steady Occlusion Event (SOE) in a scene but not a Dynamic Occlusion Event (DOE), the metrics will reflect the unsolved scenario regardless of its effectiveness in resolving the simple event.

Bash scripts are created to automate some processes:

- Execute the tracker with different configuration parameters
- Calculate tracking performance metrics with the validation tool and all generated tracking output data
- Compile all metric reports into an Excel sheet
- Evaluate the tracking results with different Multiple Object Tracking Accuracy (MOTA) error weights (steps of 0.1 from 0 to 1 in all three weights combinations).

The complete set of results (videos, clips, screenshots, reports) are accessible through the Pattern Recognition and Intelligent Systems Laboratory (PRIS-Lab) private Cloud storage at results repository.

Environment

The previous version of ACE tracking platform is available in two forms, the first one as source code maintained in PRIS-Lab private GitLab repository, the second option is a VirtualBox virtual machine image containing the source code, the development environment and the compiled tracker. The virtual

image is convenient as the environment is already set and ready to use, ACE platform still depends on a couple of old libraries that have proven to be a nuisance to set up. The environment consists of a Debian 8 "jessie" image with the required packages installed to execute the tracker: OpenCV 2.4.13.5, Yaml-Cpp 0.6.2, ALGLIB 3.15.

The development and testing of the updated ACE tracking platform was done in a Windows Subsystem for Linux (WSL) environment with Debian 11 "Bullseye" and the next libraries: OpenCV 2.4.13.5, Yaml-Cpp 0.7.0, ALGLIB 3.15. Yaml-Cpp update does not present a significant change in the code, as it is used only to parse Yaml files, which retain the same structure. The important libraries are OpenCV, to processing images and matrix operations, and ALGLIB for data processing.

Soccer player tracking with ACE tracking platform was executed in a personal computer with the following characteristics:

- OS: Debian 11 (WSL)
- CPU: AMD Ryzen 9
- GPU: RTX 3080 GDDR6 16G
- RAM: 32GB
- HD: 2 TB

During development, the cluster TARÁ from the PRIS-Lab and the Kabré supercomputer from the National Advanced Computing Collaboratory (acronym in Spanish) (CNCA) [17] were used to execute the code, particularly during the Kernel Density Estimation (KDE) development, to take advantage of a high-performance computing system in executing high costly algorithms. However, due to complications in resources availability and conflicts with code dependencies, and with the eventual availability of a local machine with enough capacity to run the code in an acceptable time, the clusters were not accessed afterwards. Following are the specifications of both cluster, as reference. It is important to note that the high-performance computing systems from both PRIS-Lab and CNCA go under frequent maintenance and upgrade, the provided information might be outdated.

Nodo	**Quantity**	**Processor**	**RAM**	**HD**	**GPU**
Master	1	Intel Xeon E5-2650	62 GB	500 GB	-
Processing	4	Intel Xeon E5-2650	251 GB	500 GB	Tesla K20m
Storage	4	Intel Xeon E5-2650	62 GB	40 TB	-

Table 5.1: TARÁ System Specification [92]

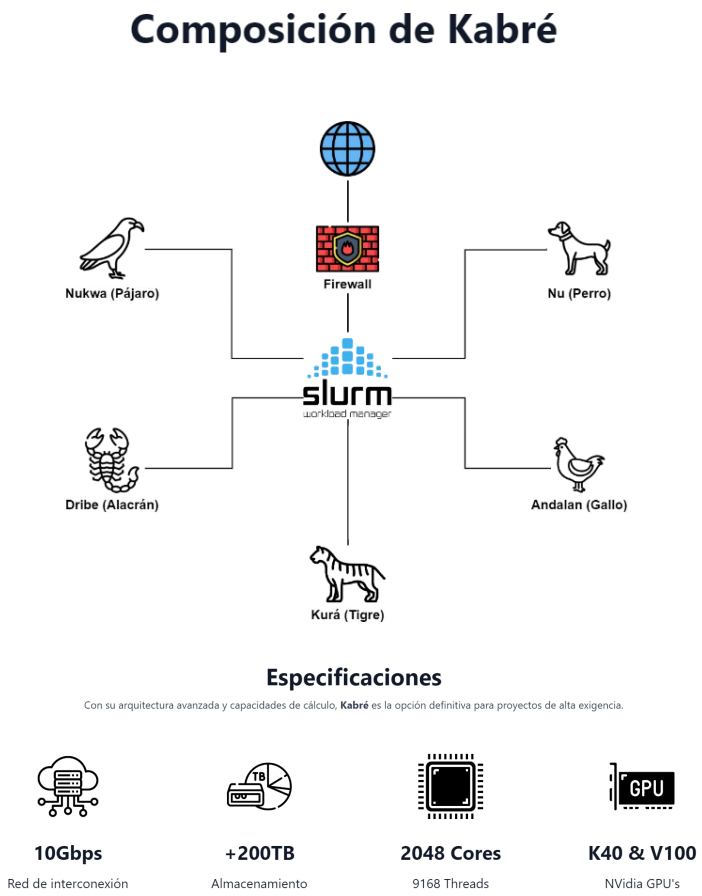


Figure 5.1: Kabre supercomputer overview [17]

The input video consist of a panoramic view result of stitching the recordings of two 4K UHD cameras, each pointing to one half of the soccer fields. The video is processed from frame number 490 to frame number 1080 for a total of 591 frames and 20 s of game. The range is selected taking into account it covers a full game play (the referees mark the start and end of the play) and that it is a dynamic scene, with occlusions of up to four players involved; a huge set of frames with no interesting interaction may not provide insightful results on the effectiveness and robustness of the algorithm when tracking players and more particularly players in occlusion events.

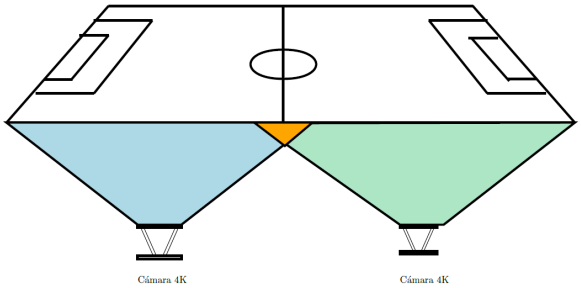


Figure 5.2: Camera placement to record soccer matches to cover the whole field [93]

ACE tracking platform configuration parameters are selected from the last status of the previous tracker version as provided in the virtual machine, see A for a full list of parameters. These parameters include thresholds and values used during the segmentation and blob creation, region of interest boundaries, batch size, player blob weights, tracking configurations, output specifications and enabling/disabling features like Multiple Multipartite Graphs (MMPG), MHT, memoirs and others as stated in 4.

The input video is selected from the PRIS-Lab repository of videos, and new ground truth annotations are generated to include player position and occlusion events. It was considered to utilize the ISSIA database videos [62], however, as stated in [92], a translator is required to parse ISSIA format to ACE format, and, following the recommendations, this database was dismissed due to the annotations incongruence, even though the database might have been cleaned up and harmonized, it was preferred to not invest time in reviewing the database but instead invest in helping to build a PRIS-Lab annotation database that can be reused with low effort by ACE and PRIS-Lab researchers and collaborators, including also information on occlusion events, which is not covered much in the reviewed literature and not part of public databases.

Figure 5.3: Example of issues with ISSIA database as informed in [93]

Figure 5.4: Example of input video recorded as part of development of ACE platform [93]

Ground Truth Data Annotation

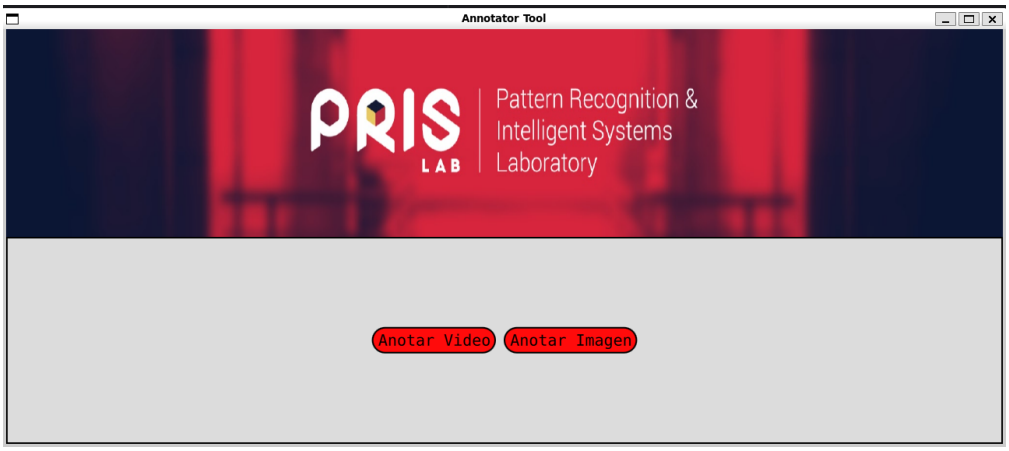
To generate the Ground Truth data sets two main tasks where executed: development of a new annotation tool and execution of an annotation campaign.

Annotation Tool

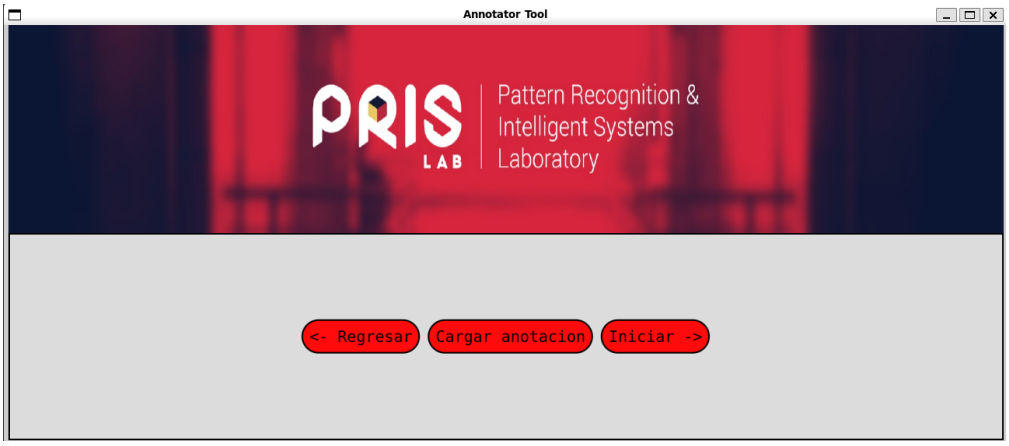
Manual annotation is a time consuming, high effort and low enjoyment task that people tend to avoid, however, it is a crucial task that generates the ground truth data sets required to validate and evaluate the algorithms. A new annotation tool with a more friendly and appealing graphical interface was developed to make annotation faster and easier. The tool is developed in Qt framework, which contains modularized C++ library classes loaded with APIs for creating graphical user interfaces and cross-platform applications [31].

The PRIS Annotation Tool is designed with a flow that lets the user select either a video or a list of images to work on, it provides the capability of loading an existent set of annotation data sets to continue on it or to create a clean new set. It provides tools like image panning, zooming in and out, annotate with points, predefined shapes or with an abstract contour made up of a succession of points. Wrong annotations can be cleared, the size of the annotation point can be increased to aid visualization, it has the capability of adding customized tags and create categories to group the annotations. It also includes a tool to merge multiple annotation files into a single one, allowing cross collaboration.

A user guide, guided example, installation executable and video tutorial are available in a public read-only directory from PRIS-Lab cloud storage. The code release 1.0 is maintained in PRIS-Lab private GitLab repository.



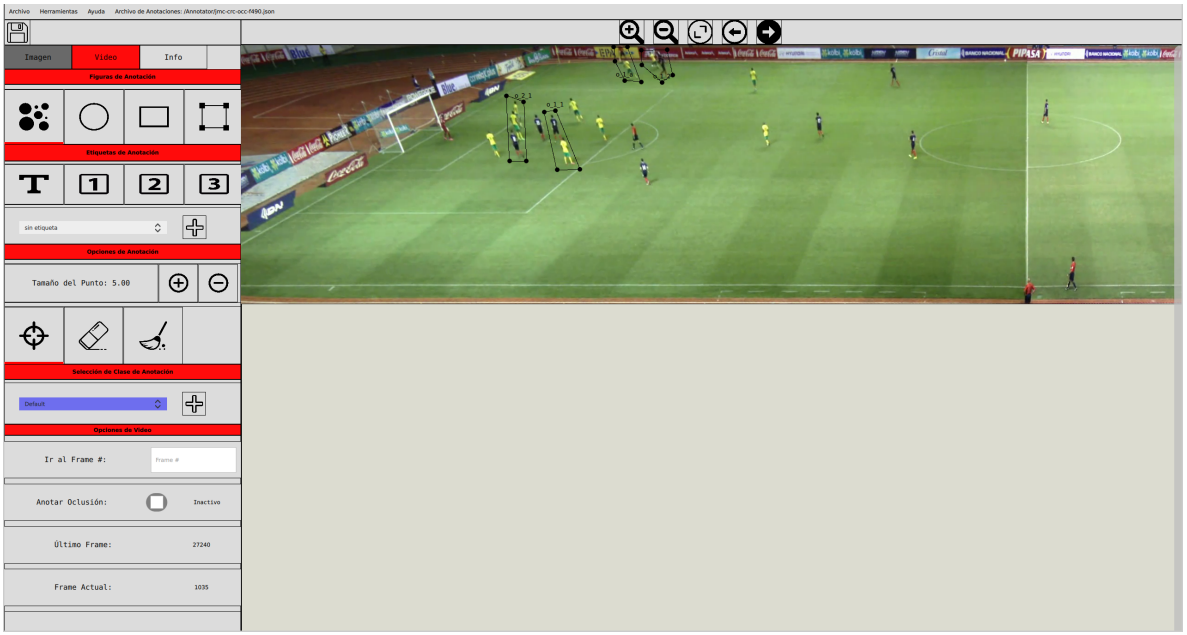
(a) Main Screen



(b) Configuration Screen

Figure 5.5: PRIS Annotation Tool

(a) Player position annotation



(b) Occlusion events annotation

Figure 5.6: PRIS Annotation Tool - Annotation pannel

Annotation Campaign

To accelerate Ground Truth generation and reduce the burden of manually annotating in a single person, an annotation campaign was designed to enroll people and distribute the annotation tasks. People were to install the PRIS Annotation Tool in their personal computers, manually annotate the assigned players and frames at their own leisure and then send the annotations via e-mail. The annotations are collected, reviewed and merged using the annotations merging tool, resulting in a single file with all the required annotations for the given frames of a video.

The participants are enrolled by filling a form created with Google Forms, the form collects a contact e-mail address, explains the purpose of the annotation campaign and asks for consent in participating, the answers are collected to a private Google Sheets document. Using Google App Scripts, an automation script is deployed to the cloud to process any forms answer, this script will check on a list of players to annotate to assign the participant the task of annotating the assigned set of players. A message is prepared with all the required information and sent to the recipient as per the form answers, include the link to the campaign resources containing the user guides, the reference files and the annotation tool installer.

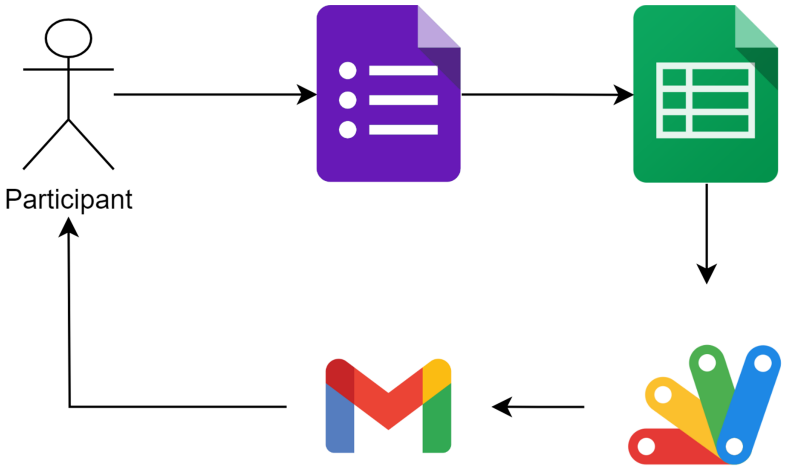
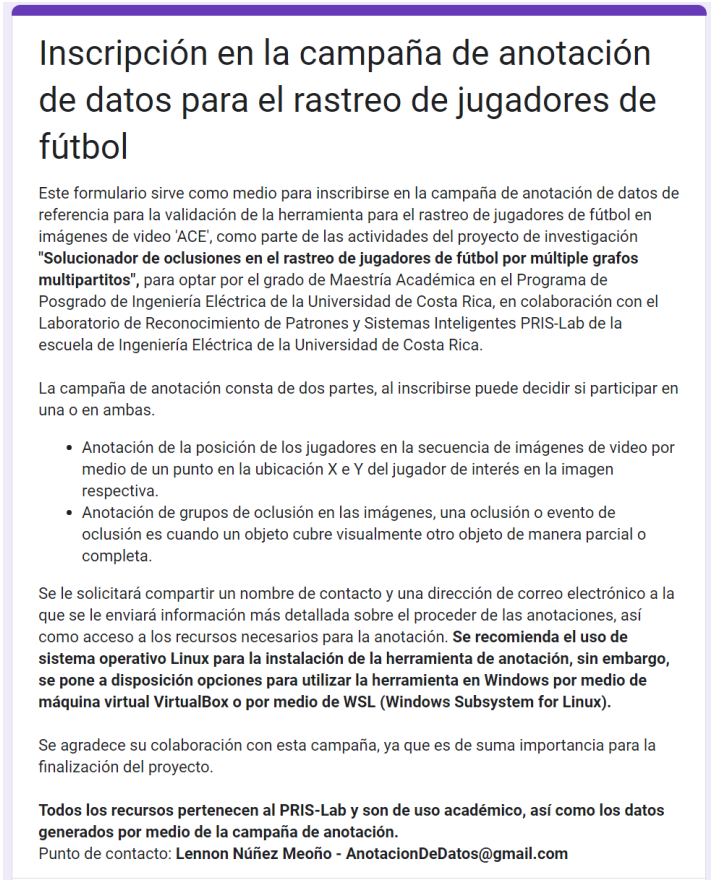


Figure 5.7: Annotation Campaign enrollment automation flow



Inscripción en la campaña de anotación de datos para el rastreo de jugadores de fútbol

Este formulario sirve como medio para inscribirse en la campaña de anotación de datos de referencia para la validación de la herramienta para el rastreo de jugadores de fútbol en imágenes de video 'ACE', como parte de las actividades del proyecto de investigación **"Solucionador de oclusiones en el rastreo de jugadores de fútbol por múltiple grafos multipartitos"**, para optar por el grado de Maestría Académica en el Programa de Posgrado de Ingeniería Eléctrica de la Universidad de Costa Rica, en colaboración con el Laboratorio de Reconocimiento de Patrones y Sistemas Inteligentes PRIS-Lab de la escuela de Ingeniería Eléctrica de la Universidad de Costa Rica.

La campaña de anotación consta de dos partes, al inscribirse puede decidir si participar en una o en ambas.

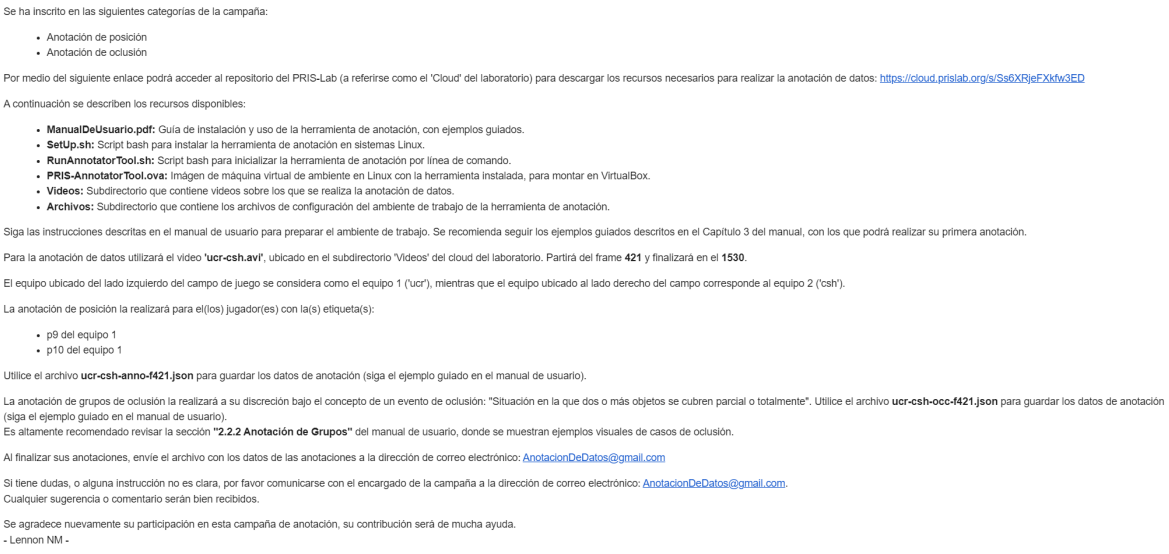
- Anotación de la posición de los jugadores en la secuencia de imágenes de video por medio de un punto en la ubicación X e Y del jugador de interés en la imagen respectiva.
- Anotación de grupos de oclusión en las imágenes, una oclusión o evento de oclusión es cuando un objeto cubre visualmente otro objeto de manera parcial o completa.

Se le solicitará compartir un nombre de contacto y una dirección de correo electrónico a la que se le enviará información más detallada sobre el proceder de las anotaciones, así como acceso a los recursos necesarios para la anotación. **Se recomienda el uso de sistema operativo Linux para la instalación de la herramienta de anotación, sin embargo, se pone a disposición opciones para utilizar la herramienta en Windows por medio de máquina virtual VirtualBox o por medio de WSL (Windows Subsystem for Linux).**

Se agradece su colaboración con esta campaña, ya que es de suma importancia para la finalización del proyecto.

Todos los recursos pertenecen al PRIS-Lab y son de uso académico, así como los datos generados por medio de la campaña de anotación.
Punto de contacto: **Lennon Núñez Meoño - AnotacionDeDatos@gmail.com**

Figure 5.8: Form to enroll into the Annotation Campaign



Se ha inscrito en las siguientes categorías de la campaña:

- Anotación de posición
- Anotación de oclusión

Por medio del siguiente enlace podrá acceder al repositorio del PRIS-Lab (a referirse como el 'Cloud' del laboratorio) para descargar los recursos necesarios para realizar la anotación de datos: https://cloud.prislab.org/s/Ss6XRjeFXkfw3ED

A continuación se describen los recursos disponibles:

- **ManualDeUsuario.pdf:** Guía de instalación y uso de la herramienta de anotación, con ejemplos guiados.
- **SetUp.sh:** Script bash para instalar la herramienta de anotación en sistemas Linux.
- **RunAnnotatorTool.sh:** Script bash para inicializar la herramienta de anotación por línea de comando.
- **PRIS-AnnotatorTool.ova:** Imágen de máquina virtual de ambiente en Linux con la herramienta instalada, para montar en VirtualBox.
- **Videos:** Subdirectorio que contiene videos sobre los que se realiza la anotación de datos.
- **Archivos:** Subdirectorio que contiene los archivos de configuración del ambiente de trabajo de la herramienta de anotación.

Siga las instrucciones descritas en el manual de usuario para preparar el ambiente de trabajo. Se recomienda seguir los ejemplos guiados descritos en el Capítulo 3 del manual, con los que podrá realizar su primera anotación.

Para la anotación de datos utilizará el video **'ucr-csh.avi'**, ubicado en el subdirectorio 'Videos' del cloud del laboratorio. Partirá del frame **421** y finalizará en el **1530**.

El equipo ubicado del lado izquierdo del campo de juego se considera como el equipo 1 ('ucr'), mientras que el equipo ubicado al lado derecho del campo corresponde al equipo 2 ('csh').

La anotación de posición la realizará para el(los) jugador(es) con la(s) etiqueta(s):

- p9 del equipo 1
- p10 del equipo 1

Utilice el archivo **ucr-csh-anno-f421.json** para guardar los datos de anotación (siga el ejemplo guiado en el manual de usuario).

La anotación de grupos de oclusión la realizará a su discreción bajo el concepto de un evento de oclusión: "Situación en la que dos o más objetos se cubren parcial o totalmente". Utilice el archivo **ucr-csh-occ-f421.json** para guardar los datos de anotación (siga el ejemplo guiado en el manual de usuario).
Es altamente recomendado revisar la sección **"2.2.2 Anotación de Grupos"** del manual de usuario, donde se muestran ejemplos visuales de casos de oclusión.

Al finalizar sus anotaciones, envíe el archivo con los datos de las anotaciones a la dirección de correo electrónico: AnotacionDeDatos@gmail.com

Si tiene dudas, o alguna instrucción no es clara, por favor comunicarse con el encargado de la campaña a la dirección de correo electrónico: AnotacionDeDatos@gmail.com.
Cualquier sugerencia o comentario serán bien recibidos.

Se agradece nuevamente su participación en esta campaña de anotación, su contribución será de mucha ayuda.
- Lennon NM -

Figure 5.9: Automatic confirmation message generated after participants enrollment into the Annotation Campaign

The volunteer is expected to follow the steps to setup PRIS-Lab Annotator Tool, load the input video and the starting point annotation file (to guarantee all participants start with the same init data), and proceed with the manual annotation. Once the task is completed, the participant sends the annotation file to the e-mail address created for the campaign, all the annotation files are collected, reviewed and merged accordingly, to generate the ground truth data set repository including the input video, a file with the position annotations and a file with the occlusion events annotation.

In spite of the efforts in advertising the annotation campaign, enrollment was low and participation even lower. Manually annotate objects is not an attractive task and it requires to dedicate several hours to the task, in the end, the generated annotations were done by a single person. The annotation campaign needs to be redesigned so as to make it more appealable to the general public; alternatives to manually annotate ground truth data set is to be taken into consideration more seriously, some options are using simulated data and integrate AI powered annotation tools like **<empty citation>**

The generated ground truth files can be accessed from the PRIS-Lab private cloud repository.

Figure 5.10: Examples of occlusion events and players' position manual annotations generated with the PRIS Annotation Tool

5.2 Results Evaluation Metrics

The following metrics are considered in evaluating the performance of the tracker: False Positives, False Negatives, True Positives, Misses error, Mismatches error, Recall, Precision, F1-Score, MOTA and MOTP. From these, focus is set on False Positives and True Positives (closely related to the segmentation and identification stages), F1-Score (harmonizes precision and recall) and MOTA (covers misses, mismatches and false positives). [11], [12]

The tracker is evaluated first by including all errors (false positives, misses, mismatches) into the weighted MOTA calculation (see 2.6), the same is later evaluated by dismissing the effect of false positives, considering these are greatly generated by the segmentation and detection stages rather than the tracking algorithm.

When comparing the results of the individual effect of each improvement feature, it can be noted that there is no much difference between the obtained MOTA values (variance of 16.3 around mean of 58.7), being the **hue filter** the one with the greatest improvement of 22% when compared with the previous tracker; F1-Score also reflects low change (variance of 4.5 around a mean of 74.4%), ranging between 73% and 78%, **MMPG** having the highest value from this set. In terms of the basic metrics, it is important to note that **hue filter** manages to reduce the False Positives in 33%, a direct result of filtering out spurious blobs with this filter. This last result is contrasted with **occlusion alarm**, **duplicate solver** and **merge splitter**, where the inclusion of faulty *Ghost Blobs* introduce new False Positives.

Feature	FP	FN	TP	MSM	Rcll	Prcn	F1	MOTA	MOTP
original	18.76	31.46	68.54	24.07	0.69	0.78	0.73	56.75	0.10
hue	12.56	29.32	70.68	20.26	0.71	0.87	0.78	69.18	0.11
mmpg	10.13	30.61	69.39	31.78	0.69	0.88	0.78	58.73	0.14
mht	17.95	30.30	69.70	24.02	0.70	0.80	0.74	58.22	0.17
oa	20.12	31.01	68.99	23.85	0.69	0.77	0.73	56.06	0.13
dp	20.32	31.01	68.99	23.17	0.69	0.78	0.73	56.91	0.13
dp_rm	20.32	31.05	68.95	23.17	0.69	0.78	0.73	56.90	0.13
ms	20.33	31.05	68.95	23.18	0.69	0.78	0.73	56.90	0.13

Table 5.2: Tracking Metrics Summary - Individual Enabled Features

Feature	FP Reduction	TP Improvement	MOTA Improvement
hue	-33.0%	3.1%	21.9%
mmpg	-46.0%	1.2%	3.5%
mht	-4.3%	1.7%	2.6%
oa	7.2%	0.7%	-1.2%
dp	8.3%	0.7%	0.3%
dp_rm	8.3%	0.6%	0.3%
ms	8.4%	0.6%	0.3%

Table 5.3: Tracking Metrics Improvement - Individual Enabled Features

When we include features combinations, e.g. MMPG with **huel filter** and **occlusion alarm**, a more varied range of results is found. For ease of comparison, the results are grouped by main feature, i.e. MHT + features, MMPG + features, and the remaining combinations. An important result is that for each group **hue filter** provides the best tracking results, which can be attributed to the reduction in False Positives.

Features	FP	FN	TP	MSM	Rcll	Prcn	F1	MOTA	MOTP
mht_hue_dp_rm_ms	10.58	28.92	71.08	19.86	0.71	0.89	0.79	71.12	0.14
mht_hue	10.60	28.58	71.42	19.89	0.71	0.89	0.79	71.10	0.14
mht_hue_dp_ms	10.58	28.89	71.11	19.94	0.71	0.89	0.79	71.04	0.14
mht_hue_dp	10.55	28.63	71.37	19.98	0.71	0.89	0.79	71.01	0.14
mht_hue_ms	10.61	28.99	71.01	20.00	0.71	0.89	0.79	70.98	0.14
mht_hue_dp_rm	10.53	29.01	70.99	20.08	0.71	0.89	0.79	70.90	0.14
mht_mmpg_hue	10.59	28.81	71.19	20.08	0.71	0.89	0.79	70.90	0.14
mht_hue_oa_ms	10.72	28.48	71.52	20.09	0.72	0.89	0.79	70.66	0.14
mht_hue_oa_dp_rm_ms	10.58	28.68	71.32	20.19	0.71	0.89	0.79	70.56	0.14
mht_hue_oa_dp_ms	10.62	28.79	71.21	20.33	0.71	0.89	0.79	70.42	0.14
mht_hue_oa_dp_rm	10.60	28.88	71.12	20.38	0.71	0.88	0.79	70.38	0.14
mht_hue_oa	10.70	28.89	71.11	20.42	0.71	0.88	0.79	70.33	0.14
mht_hue_oa_dp	10.61	28.72	71.28	20.44	0.71	0.89	0.79	70.31	0.14
mht	17.95	30.30	69.70	24.02	0.70	0.80	0.74	58.22	0.17
mht_mmpg	17.95	30.30	69.70	24.02	0.70	0.80	0.74	58.22	0.17
mht_ms	17.95	30.30	69.70	24.02	0.70	0.80	0.74	58.22	0.17
mht_dp	17.82	30.36	69.64	24.10	0.70	0.80	0.74	58.14	0.17
mht_dp_ms	17.82	30.36	69.64	24.10	0.70	0.80	0.74	58.14	0.17
mht_dp_rm	17.82	30.37	69.63	24.10	0.70	0.80	0.74	58.14	0.17
mht_dp_rm_ms	17.82	30.37	69.63	24.10	0.70	0.80	0.74	58.14	0.17
mht_oa	17.96	30.14	69.86	24.81	0.70	0.79	0.74	56.99	0.17
mht_oa_ms	17.96	30.14	69.86	24.81	0.70	0.79	0.74	56.99	0.17
mht_oa_dp	17.53	30.10	69.90	25.00	0.70	0.79	0.74	56.72	0.17
mht_oa_dp_ms	17.53	30.10	69.90	25.00	0.70	0.79	0.74	56.72	0.17
mht_oa_dp_rm	17.59	29.99	70.01	24.99	0.70	0.79	0.74	56.72	0.17
mht_oa_dp_rm_ms	17.59	29.99	70.01	24.99	0.70	0.79	0.74	56.72	0.17

Table 5.4: Tracking Metrics Summary for MHT Variations

Features	FP	FN	TP	MSM	Rcll	Prcn	F1	MOTA	MOTP
mmpg_hue_dp_ms	2.17	29.28	70.72	28.91	0.71	0.98	0.82	69.33	0.13
mmpg_hue	2.17	29.24	70.76	28.92	0.71	0.98	0.82	69.32	0.13
mmpg_hue_dp	2.17	29.41	70.59	28.92	0.71	0.98	0.82	69.32	0.13
mmpg_hue_dp_rm_ms	2.17	29.33	70.67	28.92	0.71	0.98	0.82	69.32	0.13
mmpg_hue_ms	2.17	29.20	70.80	28.92	0.71	0.98	0.82	69.32	0.13
mmpg_hue_dp_rm	2.17	29.39	70.61	28.93	0.71	0.98	0.82	69.31	0.13
mmpg_hue_oa_dp_ms	2.19	29.71	70.29	29.31	0.70	0.98	0.82	68.89	0.13
mmpg_hue_oa_dp_rm	2.18	29.65	70.35	29.31	0.70	0.98	0.82	68.89	0.13
mmpg_hue_oa_dp_rm_ms	2.19	29.66	70.34	29.31	0.70	0.98	0.82	68.89	0.13
mmpg_hue_oa_ms	2.19	29.77	70.23	29.32	0.70	0.98	0.82	68.88	0.13
mmpg_hue_oa	2.19	29.69	70.31	29.33	0.70	0.98	0.82	68.87	0.13
mmpg_hue_oa_dp	2.19	29.64	70.36	29.33	0.70	0.98	0.82	68.87	0.13
mmpg	10.13	30.61	69.39	31.78	0.69	0.88	0.78	58.73	0.14
mmpg_ms	10.13	30.61	69.39	31.78	0.69	0.88	0.78	58.73	0.14
mmpg_dp	10.13	30.59	69.41	31.79	0.69	0.88	0.78	58.72	0.14
mmpg_dp_ms	10.13	30.59	69.41	31.79	0.69	0.88	0.78	58.72	0.14
mmpg_dp_rm	10.14	30.59	69.41	31.79	0.69	0.88	0.78	58.71	0.14
mmpg_dp_rm_ms	10.14	30.59	69.41	31.79	0.69	0.88	0.78	58.71	0.14
mmpg_oa	10.05	30.88	69.12	32.4	0.69	0.88	0.77	58.03	0.14
mmpg_oa_ms	10.05	30.88	69.12	32.4	0.69	0.88	0.77	58.03	0.14
mmpg_oa_dp	10.15	30.76	69.24	32.48	0.69	0.88	0.77	57.76	0.14
mmpg_oa_dp_ms	10.15	30.76	69.24	32.48	0.69	0.88	0.77	57.76	0.14
mmpg_oa_dp_rm	10.14	30.75	69.25	32.51	0.69	0.88	0.77	57.74	0.14
mmpg_oa_dp_rm_ms	10.14	30.75	69.25	32.51	0.69	0.88	0.77	57.74	0.14

Table 5.5: Tracking Metrics Summary for MMPG Variations

Features	FP	FN	TP	MSM	Rcll	Prcn	F1	MOTA	MOTP
hue_dp_rm_ms	12.55	29.25	70.75	20.24	0.71	0.87	0.78	69.20	0.11
hue	12.56	29.32	70.68	20.26	0.71	0.87	0.78	69.18	0.11
hue_dp	12.56	29.36	70.64	20.26	0.71	0.87	0.78	69.18	0.11
hue_dp_rm	12.55	29.20	70.80	20.27	0.71	0.87	0.78	69.17	0.11
hue_dp_ms	12.59	29.43	70.57	20.26	0.71	0.87	0.78	69.14	0.11
hue_oa	12.47	29.21	70.79	20.31	0.71	0.87	0.78	69.13	0.11
hue_ms	12.63	29.41	70.59	20.28	0.71	0.87	0.78	69.12	0.11
hue_oa_ms	12.54	29.31	70.69	20.39	0.71	0.87	0.78	69.02	0.11
hue_oa_dp	12.63	29.11	70.89	20.26	0.71	0.87	0.78	68.95	0.11
hue_oa_dp_rm_ms	12.65	28.90	71.10	20.24	0.71	0.87	0.78	68.95	0.11
hue_oa_dp_ms	12.68	29.09	70.91	20.26	0.71	0.87	0.78	68.91	0.11
hue_oa_dp_rm	12.63	28.93	71.07	20.30	0.71	0.87	0.78	68.90	0.11
dp	20.32	31.01	68.99	23.17	0.69	0.78	0.73	56.91	0.13
dp_ms	20.32	31.01	68.99	23.17	0.69	0.78	0.73	56.91	0.13
dp_rm	20.32	31.05	68.95	23.17	0.69	0.78	0.73	56.90	0.13
dp_rm_ms	20.32	31.05	68.95	23.17	0.69	0.78	0.73	56.90	0.13
ms	20.33	31.05	68.95	23.18	0.69	0.78	0.73	56.90	0.13
oa	20.12	31.01	68.99	23.85	0.69	0.77	0.73	56.06	0.13
oa_ms	20.12	31.01	68.99	23.85	0.69	0.77	0.73	56.06	0.13
oa_dp_rm	20.01	30.65	69.35	24.00	0.69	0.77	0.73	55.52	0.14
oa_dp_rm_ms	20.01	30.65	69.35	24.00	0.69	0.77	0.73	55.52	0.14
oa_dp	20.01	30.92	69.08	24.05	0.69	0.77	0.73	55.45	0.14
oa_dp_ms	20.01	30.92	69.08	24.05	0.69	0.77	0.73	55.45	0.14

Table 5.6: Metrics for various feature sets

Another important aspect to highlight is the discrepancy between Precision and Multiple Object Tracking Precision (MOTP), the lowest Precision value obtained is 77%, whereas the highest MOTP is 17%. In the first case, Precision is higher as True Positives in all cases are around 70% and False Positives can become as low as 2%, we only care if the objects to be tracked are being detected, it is mainly about quantity. But in the case of MOTP we include an error from the distances between the detected objects (predicted position) and the actual objects' positions (true position), as we are tracking the centroids of the objects, the MOTP value is highly influenced by the correct centroid estimation (are we clustering all the pixels?) and the correct ground truth annotation (are we correctly annotating the players?), thus it is no surprise this metric, even though mentioned in literature, is not as popular as MOTA when presenting the results. However, such low MOTP in contrast with the rest of the metrics raises a flag into reviewing the two aforementioned aspects in evaluating the tracker.

From MMPG results (5.5) False Positives can drop to 2%, resulting in a reduction of spurious blobs of 88% comparing with the previous version of ACE when including the **hue filter** in addition with other features, **hue filter** helps by eliminating spurious blobs, which reduces the noise in the data set the tracker is to process (see figure 4.9), and the other features complement in helping the tracker better assign the identities, so missed spurious blobs are not assigned a player identity during tracking resolution. The best combination of features in this first set is MHT with **hue filter**, **duplicate solver** and **merge splitter**, however, there is no much difference between the first five combinations from table 5.4, the common features are MHT and **hue filter**.

Feature	**FP**	**FN**	**TP**	**MSM**	**Rcll**	**Prcn**	**F1**	**MOTA**	**MOTP**
original	18.76	31.46	68.54	24.07	0.69	0.78	0.73	56.75	0.10
mht_hue_dp_rm_ms	10.58	28.92	71.08	19.86	0.71	0.89	0.79	71.12	0.14

Table 5.7: Metrics for Original and mht_hue_dp_rm_ms features

Feature	**FP Red**	**TP Imp**	**MOTA Imp**
mht_hue_dp_rm_ms	-43.6%	3.7%	25.3%

Table 5.8: Performance Improvement Metrics for mht_hue_dp_rm_ms feature

Taking into account the whole set of results for the testing data set, the variance of MOTA is 38.72, the results are relatively widely spread around the mean of 63.43, if we take into account only the results of the combinations that include **hue filter**, the variance is 0.69 around a mean of 69.67, and if we exclude **hue filter** we get a variance of 5.56 around a mean of 57.7. Results including **hue**

filter are in average 20% better in MOTA than those without it, spurious filtering has the greatest impact in improving ACE tracking platform performance.

If we take out of scope the false positives error and focus on occlusion related errors (see 2.6 [56]), we can set the w_FP=0 in our MOTA calculation to get a metric that reflects only errors on players identity retention (misses and mismatches). Note in 5.9 how the improvement from including **hue filter** is reduced to 5%, which is in line with leaving out the false positives errors from the metric, **hue filter** feature is great at cleaning noisy detections, but it does not do much on its own in terms of actual tracking, as already mentioned before, it is a combination of features (cleaning the data set + handling missing identities) that generate the best results.

Feature	MOTA	FP Reduction (%)	TP Improvement (%)	MOTA Improvement (%)
original	75.93	-	-	-
rework	76.82	8.4%	0.6%	1.2%
hue	79.74	-33.0%	3.1%	5.0%
mmpg	68.22	-46.0%	1.2%	-10.2%
mht	75.98	-4.3%	1.7%	0.1%
oa	76.15	7.2%	0.7%	0.3%
dp	76.83	8.3%	0.7%	1.2%
dp_rm	76.83	8.3%	0.6%	1.2%
ms	76.82	8.4%	0.6%	1.2%

Table 5.9: MOTA evaluation with false positives error weight equals 0

When evaluating all features combinations, the best result is once again the MHT with **hue filter**, **duplicate solver** and **merge splitter**.

Feature	MOTA	MOTP	FP Red (%)	TP Imp (%)	MOTA Imp (%)
mht_hue_dp_rm_ms	80.14	0.14	-43.6%	3.7%	5.5%

Table 5.10: MOTA for the MHT with Hue Filter, Duplicate Solver and Merge Splitter features with false positives error weight equals 0

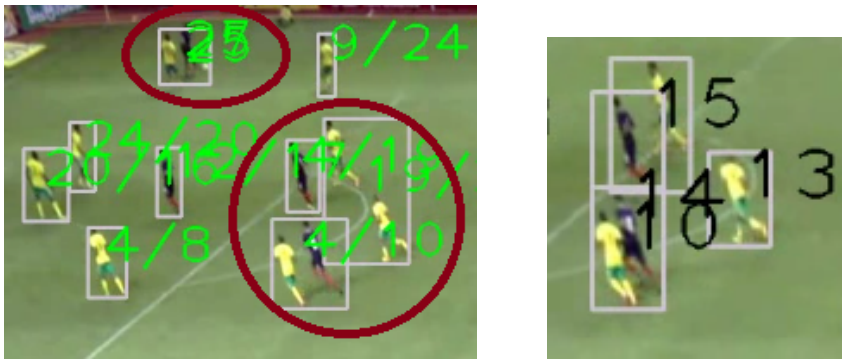
Making use of the occlusion events classification (see 2.2), we can evaluate the tracker in occlusion events identified by the OECS. Frames with simple occlusion events (CS=1) should not be a great challenge to the tracker, we can note how it succeeds in retaining the identities after an occlusion event $O^1_{581,597}$ (figure 5.11), however, it is important to note that these identity misses events start even prior to an actual occlusion event taking place (being strict in the definition by overlapping objects), this is due to errors in the segmentation of the players in which two close players are considered as one single big blob as the algorithms fail to separate them (see figure), it is good that the tracker is able

to identify this merge event and keep track of identities even after one is lost, but these errors make it more difficult to evaluate the performance on occlusion events using the traditional methods, as many of these are not actual occlusion events but just the tracker being unable to separate clusters of pixels.

(a) Frame k **(b)** Frame $k+4$ **(c)** Frame $k+6$

Figure 5.11: ACE Platform identity retention after an occlusion event

Another important take is that since we are reducing the data set when evaluating occlusion events (selection of frames from a long clip), any mistake or assertion influences more in the metrics. One example is the event $O^1_{947,1014}$, which is allocated near the top left corner of the field, even though the tracker is correctly identifying two players (tagged 25 and 27), other unresolved occlusion events like o^1_{949} and o^2_{949} will have a major impact in lowering the metrics giving a MOTA of 48%, whereas reducing the spurious blobs and recovering missing identities even if just one identity will increase the metrics, giving MOTA of 74% with MMPG, **hue filter** and **occlusion alarm** (improvement of 54%). These results are obtained without tinkering the MOTA weighted values.



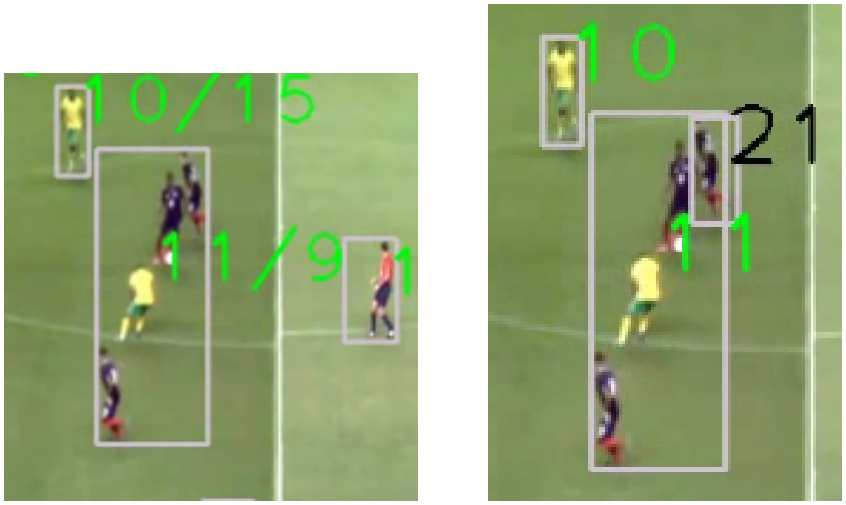
(a) Crop of multiple occlusion events with original tracker

(b) Occlusion alarm recovering one player from complex occlusion event

Figure 5.12: ACE Platform original vs updated in multiple occlusion events $O^1_{947,1014}$, o^1_{949} and o^2_{949}

Another interesting case is event $O^2_{530,534}$, the occlusion is between two players of the same team,

however, due to errors in the segmentation four players are being clustered as a single player blob. Also, as only five identities are missing in the whole scene, MOTA is quite high for this event, with a value of 80% with the unchanged tracker, and 97% with **hue filter** and **occlusion alarm** by recovering only one missing player.



(a) ACE tracker original version

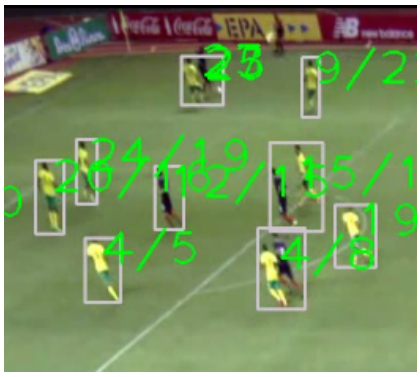
(b) ACE tracker with **hue filter** and **occlusion alarm**

Figure 5.13: Occlusion event $O^2_{530,534}$ including patients not in occlusion due to segmentation errors

When comparing scenes of more complexity, as $O^{12}_{951,979}$, involving different dynamics between the players (players switch from one occlusion region to another, making this event a composed of multiple interacting events), we can notice an improvement in the MOTA of more than 40% in all combinations of MMPG, with the highest MOTA of 76% against 50.55% from the unchanged tracker. These results are obtained without tinkering the MOTA weighted values.

(a) ACE original - Frame k

(b) ACE MHT + **hue** - Frame k



(c) ACE original - Frame k

(d) ACE MHT + **hue** - Frame k

(e) ACE original - Frame k

(f) ACE MHT + **hue** - Frame k

Figure 5.14: Snippet of complex occlusion event $O^{12}_{951,979}$ comprised of events $O^{1}_{951,974}$, $O^{1}_{957,973}$, o^{12}_{974}, $O^{3}_{975,979}$ and $O^{1}_{975,979}$. One player is recovered by the updated ACE tracker.

Features	SUTobj	FP	FN	TP	Rcll	Prcn	F1	MOTA	MOTP
mmpg_hue	521	0	26.34	73.66	0.74	1	0.85	76.07	0.17
mmpg_hue_dp	521	0	26.34	73.66	0.74	1	0.85	76.07	0.17
mmpg_hue_dp_ms	521	0	25.79	74.21	0.74	1	0.85	76.07	0.17
mmpg_hue_dp_rm	521	0	26.34	73.66	0.74	1	0.85	76.07	0.17
mmpg_hue_dp_rm_ms	521	0	25.79	74.21	0.74	1	0.85	76.07	0.17
mmpg_hue_ms	521	0	25.79	74.21	0.74	1	0.85	76.07	0.17
mmpg_hue_oa	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg_hue_oa_dp	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg_hue_oa_dp_ms	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg_hue_oa_dp_rm	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg_hue_oa_dp_rm_ms	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg_hue_oa_ms	533	0	27.31	72.69	0.73	1	0.84	75.45	0.17
mmpg	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_dp	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_dp_ms	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_dp_rm	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_dp_rm_ms	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_ms	578	1.38	24.28	75.72	0.76	0.99	0.86	75.1	0.16
mmpg_oa	597	1.34	25.93	74.07	0.74	0.99	0.85	74.21	0.17
mmpg_oa_ms	597	1.34	25.93	74.07	0.74	0.99	0.85	74.21	0.17
mmpg_oa_dp	608	1.32	25.93	74.07	0.74	0.99	0.85	74.14	0.18
mmpg_oa_dp_ms	608	1.32	25.93	74.07	0.74	0.99	0.85	74.14	0.18
mmpg_oa_dp_rm	608	1.32	25.93	74.07	0.74	0.99	0.85	74.14	0.18
mmpg_oa_dp_rm_ms	608	1.32	25.93	74.07	0.74	0.99	0.85	74.14	0.18

Table 5.11: Metrics of MMPG tracking of occlusion event $O^{12}_{951,979}$

Chapter 6

Conclusion and Recommendations

ACE tracking platform underwent multiple modifications to improve the tracking results with the occlusion events as the main target, Multipartite Graph (MPG) is robust enough to handle simple occlusion events (CS=1), the improvement in the tracking results by integrating Multiple Hypothesis Tracking (MHT) and Multiple Multipartite Graphs (MMPG) is negligible on these events. After reviewing this result, it was noted that the tracking algorithms are trying to solve identity assignments in a noisy data set with considerable presence of spurious objects, which lead to the integration of spurious filtering based on hue histograms Battacharya distance and predominant color swatches evaluation, reducing the False Positives in 33% when compared with the original tracker. Combining this spurious filter with other methods to handle occlusion events, as MHT, MMPG, **occlusion alarm** and **merge splitting**, it was possible to achieve a reduction 88% in False Positives in some instances and an increase of 25% in MOTA with a value of 71%, compared to the original tracker with a MOTA of 56.75% in the data set, result of the tracker managing to recover missing players from complex occlusion events. Moreover, when evaluating the tracker in occlusion events, it was possible to achieve a MOTA of 76%, representing an improvement of 50.1% when compared against the original tracker with a MOTA of 50.55% under the same event. If we wanted to evaluate more closely the occlusion events (taking out of the analysis the False Positives as its improvement is related to the **hue filtering**), a MOTA of 80% is achieved with MHT.

The tracker is delivered with new features and capabilities to improve tracking by reducing spurious blobs and integrating algorithms focused at recovering missing identities. A new annotation tool was developed to aid in generating ground truth annotations, an annotation campaign was designed to organize collaborative ground truth generation. ACE platform environment was adjusted to be more friendly on setup and development, and scripts are provided to automate metrics reports to ease on the tracker's evaluation.

However, the generated ground truth data set is small, there was low engagement during the annotation campaign and many segmentation errors are still present that bring noise to the tracking

algorithms. It is recommended to invest more efforts in generating a clean, complete, varied and bigger ground truth data set to sustain better the validation of future developments. The annotation campaign is to be redesigned so as to be more appealing to the general public and reduce even more the burden of manual annotation, other tools like CVAT [61] and simulated ground truth data are to be incorporated to reduce the dependency on manual annotation. ACE memoirs are a great structure to recover missing identities, it is suggested to pay more attention to this kind of methods that take historical data into resolving tracking issues. Ghost Blobs are a simple way of keeping alive missing players, but it relies strongly on the next position estimation, which is to be refactored by taking into account better ways of estimating the player's next position, Particle Filters are widely used in multiple object tracking, it could be integrated to strengthen ACE into better predicting position on missing detections. Even though **hue filtering** was quite effective in reducing the spurious blobs, the team model is created only at the beginning of the video, making it possible to become outdated easily, updating this model with new data as the batches are processed could benefit even more this filter, however, care is to be taken as to not invalidate the model by incorporating noise on updates. Lastly, ACE platform is a complex system with many configuration parameters, some improvement in the tracking results can be obtained simply by selecting the correct parameters for the correct video, it is highly recommended to invest in developing a dynamic configuration that adjusts ACE setup based on the scenes under evaluation, rather than having a static set of parameters throughout the entire processing, segmentation, detection and prunning are highly affected by the selected parameters.

Bibliography

[1] A. algorithm for tracking multiple targets - fix, "Football player tracking from multiple views - using a novel background segmentation algorithm and multiple hypothesis tracking", in *Proceedings of IEEE Transactions on Automatic Control*, vol. E98-D, 1979, pp. 423–432. DOI: 10.1109/tac.1979.1102177.

[2] *7 ways ai technology is used in soccer*, Last accessed Dec 2024, PlayMaker. [Online]. Available: https://www.playermaker.com/blog/ai-in-soccer/.

[3] *Ai camera operator repeatedly confuses bald head for soccer ball during live stream*, Last accessed Dec 2024, The Verge. [Online]. Available: https://www.theverge.com/tldr/2020/11/3/21547392/ai-camera-operator-football-bald-head-soccer-mistakes.

[4] *Ai research in soccer*, Last accessed Dec 2024, AI Soccer Hub. [Online]. Available: https://aisoccerhub.com.

[5] E. Alfaro, "Optimización del componente de inteligencia artificail del simulador simple soccer", Bachelors Degree graduation project, M.S. thesis, Universidad de Costa Rica, Sep. 2049.

[6] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis, and G. Karaseitanidis, "Multiple hypothesis tracking implementation", in (Laser Scanner Technology), Laser Scanner Technology. IntechOpen, 2012, ch. 10.

[7] J. Amin, D. Thakore, and N. Patel, "Object tracking: A survey", *International Journal for Research in Applied Science and Engineering Technology*, vol. 6, no. III, pp. 3455–3461, 2018.

[8] Y. Bar-Shalom, *Multitarget-Multisensor Tracking: Advanced Applications*. ArtechHouse, 096483122, Nonvood, MA, 1990.

[9] L. Bazzani, D. Bloisi, and V. Murino, *A comparison of multi hypothesis kalman filter and particle filter for multi-target tracking*, 2009.

[10] T. Bebie and H. Bieri, "Soccerman-reconstructing soccer games from video sequences", in *1998 International Conference on Image Processing. ICIP98*, 1998. DOI: 10.1109/ICIP.1998.723665.

[11] K. Bernardin and R. Stiefelhagen, "Evaluatingmultiple object tracking performance: The clear motmetrics", *EURASIP Journal on Image and Video Processing*, vol. 2008, 2008. DOI: 10.1155/2008/246309.

[12] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment", in *Proceedings of the Sixth IEEE International Workshop on Visual Surveillance*, IEEE, 2006, pp. 149–156. DOI: 10.1109/IVS.2006.118.

[13] S. Blackman, "Multiple hypothesis tracking for multiple target tracking", in *IEEE Aerospace and Electronic Systems Magazine*, vol. 19(1), 2004s, pp. 5–18. DOI: 10.1109/maes.2004.1263228.

[14] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multi-person tracking-by-detection from a single uncalibrated camera", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, 2011, pp. 1820–1833. DOI: 10.1109/TPAMI.2010.232.

[15] F. Clemente, J. Sequeiros, A. Correira, F. Sylva, and F. Laurenço, "Measuring the dispersion of the players", in (SpringerBriefs in Applied Sciences and Technologies), SpringerBriefs in Applied Sciences and Technologies. Springer, 2018, pp. 43–53. DOI: doi.org/10.1007/978-3-319-59029-5_5.

[16] F. M. Clemente, M. S. Couceiro, F. M. L. Martins, and R. Mendes, "An online tactical metrics applied to football game", *Research Journal of Applied Sciences, Engineering and Technology*, vol. 5(5), pp. 1700–1719, 2013. DOI: 10.19026/rjaset.5.4926.

[17] CNCA/CeNAT, *Kabré*, Last accessed Dec 2024. [Online]. Available: https://kabre.cenat.ac.cr.

[18] I. J. Cox and S. Hingorani, "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18(2), 1996, pp. 138–150. DOI: 10.1109/34.481539.

[19] J. Czyz, B. Risti, and B. Macq, "A color-based particle filter for joint detection and tracking of multiple objects", in *IEEE International Conference in Acoustics, Speech, and Signal Processing*, vol. 2, 2005.

[20] A. Dearden, Y. Demiris, and O. Grau, "Tracking football player movement form a single moving camera using particle filters", in *CVMP-2006*, 2006, pp. 29–37.

[21] Y. Dodge, *The Oxford Dictionary of Statistical Terms*. Oxford University Press, 2003, ISBN: 978-0-19-920613-1.

[22] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later", in *The Oxford Handbook of non-linear filtering*. OUP Oxford, 2012, pp. 656–704. DOI: 10.1.1.157.772.

[23] FIFA, *Dispositivos de seguimiento electrónico del rendimiento*, Last accessed July 2019. [Online]. Available: https://football-technology.fifa.com/es/media-tiles/epts1/.

[24] *Fifa electronic performance and tracking systems*, Last accessed Dec 2024, FIFA. [Online]. Available: https://inside.fifa.com/technical/football-technology/standards/epts.

[25] P. Figueroa, N. Leite, and R. M. L. Barros, "Tracking soccer players aiming their kinematical motion analysis", *Computer Vision and Image Understanding*, vol. 101, pp. 122–135, 2006. DOI: 10.1016/j.cviu.2005.07.006.

[26] P. Figueroa, N. Leite, R. M. L. Barros, I. Cohen, and G. Medioni, "Tracking soccer players using the graph representation", in *17th International Conference on Pattern Recognition*, vol. 4, 2004.

[27] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multiple object tracking using k-shortest paths optimization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1800–1808, 2006. DOI: 10.1109/TPAMI.2006.238.

[28] H. Folgado, K. Lemmink, W. Frencken, and J. Sampaio, "Length, width and centroid distance as measures of teams tactical performance in youth football", *European Journal of Sport Science*, vol. 14(1), pp. 487–492, 2012. DOI: 10.1080/17461391.2012.730060.

[29] E. Fortunato, W. Kreamer, S. Mori, C. Chee-Yee, and G. Castanon, "Generalized murty's algorithm with application to multiple hypothesis tracking", in *Proceedings of 2007 10th International Conference on Information Fusion*, 2007. DOI: 10.1109/icif.2007.4408017.

[30] S. Gedikli, J. Bandouch, N. Hoyningen-Huene, B. Kirchlechner, and M. Beetz, "An adaptive vision system for tracking soccer players from variable camera settings", in *Fifth International Conference on Computer Vision Systems*, 2007.

[31] Q. Group, *Qt framework*, Last accessed Dec 2024. [Online]. Available: https://www.qt.io/product/framework.

[32] R. Hess and A. Fern, "Discriminatively trained particle filters for complex multi-object tracking", in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 240–247. DOI: 10.1109/CVPR.2009.5206801.

[33] B. K. P. Horn and B. G. Schunck, "Determining optical flow", *Artificial Intelligence*, vol. 17, pp. 185–203, 1981. DOI: 10.1016/0004-3702(81)90024-2.

[34] Z. L. Inc., *Zepp play - soccer*, Last accessed Jan 2020. [Online]. Available: http://www.zepp.com/en-us/soccer/.

[35] H. Itoh, T. Takiguchi, and Y. Ariki, "3d tracking of soccer players using time-situation graph in monocular image sequence", in *21st International Conference on Pattern Recognition*, 2012.

[36] S. Iwase and H. Saito, "Parallel tracking of all soccer players by integrating detected positions in multiple view images", Department of Information and Computer Science, Keio University, Japan, Tech. Rep., 2004.

[37] JetBrains, *Clion a cross-platform ide for c and c++*, Last accessed Dec 2024. [Online]. Available: https://www.jetbrains.com/clion/.

[38] S. Jiang H. Fels and J. Little, "A linear programming approach for multiple object tracking", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. DOI: 10.1109/cvpr.2007.383180.

[39] S.-W. Joo and R. Chellapa, "A multiple-hypothesis approach for multiobject visual tracking", in *IEEE Trans. Image Process*, vol. 16(11), 2007, pp. 2849–2854. DOI: 10.1109/tip.2007.906254.

[40] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. DOI: 10.1109/iccv.2015.533.

[41] Kizanaro, *Kizanaro*, Last accessed May 2019. [Online]. Available: https://cie.ort.edu.uy/35730/18/kizanaro.html.

[42] A. Koutsia, N. Grammalidis, K. Dimitropoulos, M. Karaman, and L. Goldmann, "Football player tracking from multiple views - using a novel background segmentation algorithm and multiple hypothesis tracking", in *Proceedings of the Second International Conference on Computer Vision Theory and Applications*, INSTICC, SciTePress, 2007, pp. 523–526. DOI: 10.5220/0002051205230526.

[43] M. Kristan, J. Pers, M. Perse, and S. Kovacic, "Closed-world tracking of multiple interacting targets for indoor-sports applications", in *Computer Vision and Image Understanding*, vol. 113, 2009, pp. 598–611. DOI: 10.1016/j.cviu.2008.01.009.

[44] C. D. Kuglin and D. A. Hines, "The use of multiple hypothesis tracking for vehicle surveillance", in *Proceedings of the IEEE Conference on Decision and Control*, 1975.

[45] B. Y. Lee, L. H. Liew, W. S. Cheah, and Y. C. Wang, "Occlusion handling in video object tracking: A survey", *IOP Conference Series: Earth and Environmental Science*, vol. 18, p. 012 097, 2014. DOI: 10.1088/1755-1315/18/1/012097.

[46] S. Lefèvre, C. Fluck, B. Maillard, and N. Vincent, "A fast snake-based method to track football players", in *IAPR Workshop on Machine Vision Applications*, The University of Tokyo, Japan, 2000.

[47] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context", *arXiv preprint arXiv:1405.0312*, 2014. [Online]. Available: https://arxiv.org/abs/1405.0312.

[48] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T. Kim, "Multiple object tracking: A literature review", *Artificial Intelligence*, vol. 293, 2021. DOI: 10.1016/j.artint.2020.103448.

[49] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "A survey on player tracking in soccer videos", *Computer Vision and Image Understanding*, vol. 159, pp. 19–46, 2017. DOI: 10.1016/j.cviu.2017.02.002.

[50] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "Appearance-based multiple hypothesis tracking: Application to soccer broadcast videos analysis", *Signal Processing: Image Communication*, vol. 55, pp. 157–170, 2017. DOI: 10.1016/j.image.2017.04.001.

[51] MatchAnalysis, *Match analysis – unrivaled video and statistical analysis for soccer (football)*, Last accessed May 2019. [Online]. Available: http://matchanalysis.com/index.htm.

[52] I. McHale and P. Scarf, "Modelling soccer matches using bivariate discrete distributions with general dependence structure", *Statistica neerlandica*, vol. 61(4), pp. 432–445, 2007. DOI: doi.org/10.1111/j.1467-9574.2007.00368.x.

[53] T. Misu, M. Naemura, W. Zheng, Y. Izumi, and K. Fukui, "Robust tracking of soccer players based on data fusion", in *16th International Conference on Patter Recognition*, vol. 1, 2002.

[54] A. Mora, "Cell phenotype classification using m-phase features in live-cell bright field time-laps microscopy", M.S. thesis, Universidad de Costa Rica, 2018.

[55] E. Morais, A. Ferreira, S. A. Cunha, R. M. Barros, A. Rocha, and S. Goldenstein, "A multiple camera methodology for automatic localization and tracking of futsal players", *Pattern Recognition Letters*, vol. 39, pp. 21–30, 2014.

[56] *Motchallenge: The multiple object tracking benchmark*, Last accessed Dec 2024, MOTChallenge. [Online]. Available: https://motchallenge.net.

[57] G. N.J., S. D.J., and S. A.F.M., "Novel approach to nonlinear/non-gaussian bayesian state estimation", in *F – Radar and signal Processing*, vol. 140, 1993, pp. 107–113. DOI: 10.1049/ip-f-2.1993.0015.

[58] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "An adaptive color-based particle filter", *Image and Vision Computing*, vol. 21(1), pp. 99–110, 2003. DOI: doi.org/10.1016/S0262-8856(02)00129-4.

[59] L. Núñez-Meoño, M. Villalta, and F. Siles-Canales, "Initial approachon soccer match'sscene classification by players'field spatial distribution", *Tecnología en Marcha*, vol. 33(5), pp. 60–65, 2020. DOI: doi.org/10.18845/tm.v33i5.5077.

[60] H.-W. Ok, Y. Seo, and K.-S. Hong, "Multiple soccer players tracking by condensation with occlusion alarm probability", IIP Lab., Pohang University of Science, and Technology (POSTECH), Republic of Korea, Tech. Rep., 2002.

[61] *Open data annotation platform*, Last accessed Dec 2024, CVAT. [Online]. Available: https://www.cvat.ai.

[62] T. D. Orazio, M. M.Leo, P. N., and P. Mazzeo, "A semi-automatic system for ground truth generation of soccer video sequences", in *6th IEEE International Conference on Advanced Video and Signal Surveillance*, 2009.

[63] *Paint, sports analysis tool*, Last accessed Aug 2019, ChyronHego. [Online]. Available: https://www.thebroadcastbridge.com/content/entry/11407/chyronhego-paint-7.4-provides-powerful-sports-telestration-and-analysis.

[64] N. S. Paridhi Swaroop, "An overview of various template matching methodologies in image processing", *International Journal of Computer Applications*, vol. 153, no. 10, pp. 8–14, Nov. 2016, ISSN: 0975-8887. DOI: 10.5120/ijca2016912165. [Online]. Available: https://ijcaonline.org/archives/volume153/number10/26437-2016912165/.

[65] P. Petsas and P. Kaimakis, "Soccer player tracking using particle filters", in *2016 IEEE International Symposium on Signal Processing and Information Technology*, 2016.

[66] *Playertv: Advanced player tracking and identification for automatic soccer highlight clips*, Last accessed Dec 2024, PlayMaker. [Online]. Available: https://www.playermaker.com/blog/ai-in-soccer/.

[67] *Playmaker*, Last accessed Dec 2024, PlayMaker. [Online]. Available: https://www.playermaker.com/teams/.

[68] J. M. O. R. D. Del C. Soler, "Multiple hypothesis tracking for object tracking in dynamic environments", in *IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 2254–2259.

[69] R. Radakovic, M. Dopsaj, and R. Vulovic, "The reliability of motion analysis of elite soccer players during match measured by the tracking motion software system", in *2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE)*, 2015. DOI: 10.1109/BIBE.2015.7367676.

[70] D. B. Reid, "An algorithm for tracking multiple targets", in *IEEE Transactions on Automatic Control*, vol. 24, IEEE, 1979, pp. 843–854. DOI: 10.1109/TAC.1979.1102136.

[71] E. Ristani and C. Tomasi, "Features for multi-target tracking: A comparative study", in *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[72] J. Rollin, R. Giulianotti, P. Christopher, R. Weil, and B. Joy, *Football soccer*, Last accessed May 2019. [Online]. Available: https://www.britannica.com/sports/football-soccer/Play-of-the-game.

[73] K. H. Rosen, *Discrete Mathematics and Its Applications*, 5th ed. McGraw-Hill, 2003.

[74] H. Sabirin, H. Sankoh, and S. Naito, "Automatic soccer player tracking in single camera with robust occlusion handling", in *IEICE Transaccions on Information and Systems*, vol. E98-D, 2015, pp. 1580–1588. DOI: 10.1587/transinf.2014edp7313.

[75] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system", in *2016 6th International Conference on System Engineering and Technology*, 2016.

[76] R. Sałustowicz, M. Wiering, and J. Schmidhuber, "Learning team strategies: Soccer case studies", *Machine Learning*, vol. 33, pp. 263–282, 1998. DOI: doi.org/10.1023/A:1007570708568.

[77] M. Sanjeev, B. Ristic, N. Gordon, and T. Mansell, "Bearings-only tracking of manoeuvering targets using particle filters", *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2351–2365, 2004.

[78] *Semi-automated offside technology*, Last accessed Dec 2024, FIFA. [Online]. Available: https://inside.fifa.com/technical/football-technology/football-technologies-and-innovations-at-the-fifa-world-cup-2022/semi-automated-offside-technology.

[79] Y. Seo, S. Choi, H. Kim, and K.-S. & Hong, "Where are the ball and players? soccer game analysis with color-based tracking and image mosaick", in *9th International Conference on Image Analysis and Processing*, vol. II, Springer-Verlag, 1997, pp. 196–203.

[80] H. B. Shitrit, J. Berclaz, and F. Fleuret, "Multi-commodity network flow for tracking multiple people", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, 2014, pp. 196–203. DOI: 10.1109/tpami.2013.210.

[81] F. Siles, "Automated semantic annotation of football games from tv broadcast", Ph.D. dissertation, Technische Universitad Munchen, 2014.

[82] F. Siles and J. C. Saborío, "Parallel spatial segmentation for the automated analysis of football", in *5th IEEE International Workshop and Conference on Bioinspired Intelligence*, 2015.

[83] R. Singer, R. Sea, and K. Housewright, "Derivation and evaluation of improved tracking filter for use in dense multitarget environments", in *IEEE Trans. Information Theory*, vol. 20(4), 1974, pp. 423–432. DOI: 10.1109/tit.1974.1055256.

[84] J. Smits, E. D. Dickson, and L. H. Matthies, "A comparison of two approaches to multiple hypothesis tracking", *International Journal of Computer Vision*, vol. 108, no. 3, pp. 272–284, 2014. DOI: 10.1007/s11263-014-0709-x.

[85] H. M. Solberg, M. H. Sarkhoosh, S. Gautam, S. S. Sabet, P. Halvorsen, and C. Midoglu, *Playertv: Advanced player tracking and identification for automatic soccer highlight clips*, 2024. DOI: 10.48550/arXiv.2407.16076. arXiv: 2407.16076 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2407.16076.

[86] S. Soomro K. Khokhar and M. Shah, "Tracking when the camera looks away", in *2015 IEEE International Conference on Computer Vision Workshop*, 2015.

[87] W. Spearman, A. Basye, G. Dick, R. Hotovy, and P. Pop, "Physics-based modeling of pass probabilities in soccer", in *MIT Sloan Sports Analytics Conference*, 2017.

[88] M.-L. Sport, *Mc-lloyd products*, Last accessed Dec 2024. [Online]. Available: https://mclloyd.com/home/gps-preparation-physique/.

[89] STATS, *Stats player tracking*, Last accessed May 2019. [Online]. Available: https://www.stats.com/player-tracking/.

[90] STATS, *Stats sportvu football player tracking*, Last accessed May 2019. [Online]. Available: https://www.stats.com/sportvu-football/.

[91] *Tracab optical tracking, product information sheet*, Last accessed May 2019, ChyronHego. [Online]. Available: https://chyronhego.com/wp-content/uploads/2019/01/TRACAB-PI-sheet.pdf.

[92] M. Villalta, "Football players tracking using multipartite graphs with ultra high definition video", M.S. thesis, Universidad de Costa Rica, 2019.

[93] M. Villalta and F. Siles, "Parallelization of a multipartite graph matching algorithm for tracking multiple football players", in *Fifth International Conference on Parallel, Distributed and Grid Computing*, 2018. DOI: 10.1109/pdgc.2018.8745720.

[94] M. Waskom, *An introduction to seaborn*, Last accessed June 2020. [Online]. Available: https://seaborn.pydata.org/introduction.html.

[95] S. Węglarczyk, "Kernel density estimation and its application", in *ITM Web of Conferences, XLVIII Seminar of Applied Mathematics*, vol. 23(00037), 2013, pp. 1700–1719. DOI: 10.1051/itmconf/20182300037.

[96] L. Wei-Lwun, T. Jo-Anne, J. Little, and K. Murphy, "Learning to track and identify players from broadcast sports videos", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, 2019.

[97] R. Wood, *World's most popular sports by fans*, Last accessed May 2019, Topend Sports. [Online]. Available: https://www.topendsports.com/world/lists/popular-sport/fans.htm.

[98] M. Xu, J. Orwell, and G. Jones, "Tracking football players with multiple cameras", Digital Imaging Research Centre, Kingston University, Kingston upon Thames, KT1 2EE, UK, Tech. Rep., 2004.

[99] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey", *ACM Computing Surveys*, vol. 38, no. 13, pp. 1–16, 2004. DOI: 10.1145/1177352.1177355.

[100] K. Yoon, Y. Song, and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views", in *IET Image Processing*, vol. 12(7), 2018, pp. 1175–1184.

[101] D. A. Zaugg, A. A. Samuel, D. E. Waagen, and H. A. Schmitt, "A comparison of particle filters and multiple-hypothesis extended kalman filters for bearings-only tracking", in *Acquisition, Tracking, and Pointing XVIII*, vol. 5430, 2004. DOI: 10.1117/12.541625.

Appendices

Appendix A

ACE tracking platform configuration file

```
# Segmentation parameters
segmentator:
  mode: 1
  pris:
    hue_min: 60
    hue_max: 180
    spurious_percentage: 10
    area_max_threshold: 20000
    kernel_size: 9
    hue_threshold: 93
    enable_morph_line_remover: 1
    hmorph_axis: 30
    vmorph_axis: 2
    enable_hough_line_remover: 1
    line_kernel: 10
    minLineLength: 50
    maxLineGap: 10
    shape_contour_remover: 9
    lower_boundary: 10
    upper_boundary: 95
    enable_boundary_blob_remover: 0
    enable_morph_operations: 1
    morph_erode_kernel: 1
    morph_dilate_kernel: 5
    enable_manual_field_selection: 1
    enable_white_remover: 0
    top_left_point_x: 871
    top_left_point_y: 42
    top_right_point_x: 2757
    top_right_point_y: 40
```

```
    bottom_left_point_x: 20
    bottom_left_point_y: 524
    bottom_right_point_x: 3587
    bottom_right_point_y: 523
    enable_blob_hue_prune: 1
    enable_blob_predominant_color_prune: 1
    predominant_color_prune_no_of_colors: 4
    predominant_color_prune_black_threshold: 20

  mog:
    history: 100
    varThreshold: 16
    bShadowDetection: 0
    learningRate: 13
    spurious_percentage: 4
    gaussian_kernel: 3

# Multipartite graph tracker parameters
mpgt:
  start_frame: 490
  stop_frame: 1080
  graph_window: 50
  position_weight: 0.08
  histogram_weight: 8.0
  contour_weight: 1.0
  area_weight: 0.0025
  speed_weight: 0.05
  direction_weight: 2500.0
  depth_weight: 0.08
  max_speed: 600
  max_merge_distance: 100
  min_blob_area_threshold: 30
  merged_blob_area_threshold: 150
  init_tkp_threshold: 25
  enable_occlusion_solver: 4
  enable_abduction_finder: 1
  blob_memoir_length: 300
  memoir_diff_threshold: 999999
```

```
  enable_clustering: 0
  hue_clustering_only: 0
  how_many_colors: 4
  kmean_colors_cuantization: 16
  area_as_cluster_feature: 0
  clustering_mode: 4
  n_clusters: 6
  match_threshold: 50
  match_method: 2
  enable_kde: 0
  enable_pdl: 0
  enable_mmpg: 1
  enable_mht: 0
  enable_occlusion_alarm: 0
  enable_duplicate_solver: 0
  enable_duplicate_solver_rematch_tags: 0
  enable_merge_split: 0

# Drawer settings
drawer:
  enable_boundary_blob_remover: 0
  lower_boundary: 10
  upper_boundary: 90
  draw_tags: 1
  draw_contours: 0
  draw_bounding_box: 1
  draw_blob_ids_also: 0
  draw_cluster_ids_also: 0
  draw_trajectories: 0
  font_scale: 3
  bounding_box_thickness: 2
  trajectories_radius: 2
  enable_color_balance: 1
  color_balance_percent: 1
  draw_pdl: 1
  draw_tag_names_also: 0
  draw_frame_number: 1
```

```
# PDL settings
pdl:
  top_left_point_x: 871
  top_left_point_y: 42
  top_right_point_x: 2757
  top_right_point_y: 40
  bottom_left_point_x: 20
  bottom_left_point_y: 524
  bottom_right_point_x: 3587
  bottom_right_point_y: 523
  threshold_level: "MID"

# Occlusion
occlusion:
  proximity_weight_size_ratio_width: 0.5
  proximity_weight_size_ratio_height: 0.5
  proximity_weight_distance: 1
  proximity_threshold: 100
  distance_threshold: 70
  overlap_percentage_threshold: 0
  resolve_proximity_threshold: 40

# Duplicate Solver
duplicate_solver:
  proximity_threshold: 100
  proximity_weight_size_ratio_width: 0.5
  proximity_weight_size_ratio_height: 0.5
  proximity_weight_distance: 1
```

Appendix B

Annotation Campaign Notifications Google App Scripts Code

```
function getPlayerTag() {
  var sheet = SpreadsheetApp.getActive().getSheetByName('EtiquetasAsignadas');
  var data = sheet.getDataRange().getValues();

  var foundTag = false;
  var tag;

  for (var i = 1; i < data.length; i++)  {
    if (data[i][1] == "No") {
      tag = data[i][0];
      sheet.getRange(i+1,2).setValue("Yes");
      sheet.getRange(i+1,3).setValue(data[i][2]+1);
      foundTag = true;
      Logger.log("Modified availability of tag " + data[i][0] + " to value Yes");
      return [tag, sheet.getRange(i+1,4).getValue()];
    }
  }

  if (!foundTag) {
    for (var i = 1; i < data.length; i++)  {
      var cell = sheet.getRange(i+1,2);
      cell.setValue("No");
      foundTag = false;
      Logger.log("Modified availability of tag " + data[i][0] + " to value No");
    }
    return getPlayerTag();
  }
}
```

```
function sendEmail(e) {
  var personName = e.values[1];
  var recipient = e.values[2];
  var anno = e.values[3] == "Sí"? "<li>Anotación de posición</li>" : "";
  var occ = e.values[4] == "Sí"? "<li>Anotación de oclusión</li>" : "";

  //var video = "jmc-crc-fps30.avi";
  //var anno_file = "jmc-crc-anno-f490.json";
  //var occ_file = "jmc-crc-occ-f421.json";
  //var f0 = 409;
  //var ff = 1080;
  //var team1 = "jmc";
  //var team2 = "crc";

  var video = "ucr-csh.avi";
  var f0 = 421;
  var ff = 1530;
  var team1 = "ucr";
  var team2 = "csh";
  var anno_file = "ucr-csh-anno-f421.json";
  var occ_file = "ucr-csh-occ-f421.json";

  var players_per_answer = 2;
  tag_team = [];
  var tag = [];
  var team = [];
  for (var times = 1; times <= players_per_answer; times++) {
    tag_team = getPlayerTag()
    tag.push(tag_team[0])
    team.push(tag_team[1])
  }

  var msjAnnoInst = "<p>La anotación de posición la realizará para el(los) jugador(es) con la(s) etiqueta(s)
       : ";
  for (var i = 0; i < tag.length; i++) {
    msjAnnoInst += "<li>" + tag[i] + (team[i] == 0 ? "" : " del equipo " + team[i]) + "</li>";
  }
  msjAnnoInst += "Utilice el archivo <b>" + anno_file + "</b> para guardar los datos de anotación (siga el
       ejemplo guiado en el manual de usuario).</p>";
```

```
var msjOccInst =  "<p>La anotación de grupos de oclusión la realizará a su discreción bajo el concepto de
     un evento de oclusión: "
  + "\"Situación en la que dos o más objetos se cubren parcial o totalmente\". Utilice el archivo <b>" +
       occ_file + "</b> para guardar los datos de anotación (siga el ejemplo guiado en el manual de
       usuario)."
  + "<br>Es altamente recomendado revisar la sección <b>\"2.2.2 Anotación de Grupos\"</b> del manual de
       usuario, donde se muestran ejemplos visuales de casos de oclusión.</p>";

var annoInst = e.values[3] == "Sí"? msjAnnoInst : "";
var occInst = e.values[4] == "Sí"? msjOccInst : "";

var message = {
  to: recipient,
  subject: "Inscripción en la Campaña de Anotación de Datos",
  htmlBody: "<p>" + personName + ", gracias por inscribirse en la campaña de anotación de datos para el
       rastreo de jugadores de fútbol,</p>"
            + "<p>Se ha inscrito en las siguientes categorías de la campaña: <ul>" + anno + occ + "</ul></
                p>"
            + "<p>Por medio del siguiente enlace podrá acceder al repositorio del PRIS-Lab (a referirse
                como el 'Cloud' del laboratorio) para descargar "
            + "los recursos necesarios para realizar la anotación de datos: https://cloud.prislab.org/s/
                Ss6XRjeFXkfw3ED</p>"
            + "<p>A continuación se describen los recursos disponibles: <ul>"
            + "<li><b>ManualDeUsuario.pdf:</b> Guía de instalación y uso de la herramienta de anotación,
                con ejemplos guiados.</li>"
            + "<li><b>SetUp.sh:</b> Script bash para instalar la herramienta de anotación en sistemas
                Linux.</li>"
            + "<li><b>RunAnnotatorTool.sh:</b> Script bash para inicializar la herramienta de anotación
                por línea de comando.</li>"
            + "<li><b>PRIS-AnnotatorTool.ova:</b> Imágen de máquina virtual de ambiente en Linux con la
                herramienta instalada, para montar en VirtualBox.</li>"
            + "<li><b>Videos:</b> Subdirectorio que contiene videos sobre los que se realiza la anotación
                de datos.</li>"
            + "<li><b>Archivos:</b> Subdirectorio que contiene los archivos de configuración del ambiente
                de trabajo de la herramienta de anotación.</li>"
            + "</ul></p>"
            + "<p>Siga las instrucciones descritas en el manual de usuario para preparar el ambiente de
                trabajo."
```

```
            + " Se recomienda seguir los ejemplos guiados descritos en el Capítulo 3 del manual, con los
                 que podrá realizar su primera anotación.</p>"
            + "<p>Para la anotación de datos utilizará el video <b>'" + video + "'</b>, ubicado en el
                 subdirectorio 'Videos' del cloud del laboratorio."
            + " Partirá del frame <b>" + f0 + "</b> y finalizará en el <b>" + ff + "</b>."
            + "<p> El equipo ubicado del lado izquierdo del campo de juego se considera como el equipo 1
                 ('" + team1 + "'), "
            + " mientras que el equipo ubicado al lado derecho del campo corresponde al equipo 2 ('" +
                 team2 + "'). </p>"
            + annoInst + occInst
            + "<p>Al finalizar sus anotaciones, envíe el archivo con los datos de las anotaciones a la
                 dirección de correo electrónico: AnotacionDeDatos@gmail.com</p>"
            + "<p>Si tiene dudas, o alguna instrucción no es clara, por favor comunicarse con el encargado
                  de la campaña a la dirección de "
            + "correo electrónico: AnotacionDeDatos@gmail.com.<br>Cualquier sugerencia o comentario serán
                 bien recibidos.</p>"
            + "<p>Se agradece nuevamente su participación en esta campaña de anotación, su contribución
                 será de mucha ayuda."
            + "<br>- Lennon NM -</p>"
            ,
   replyTo: "AnotacionDeDatos@gmail.com",
   name: "AnotacionDeDatos",
   cc: "AnotacionDeDatos@gmail.com"
 }

 MailApp.sendEmail(message);
 Logger.log("Mensaje de inscripcion enviado a la dirección: " + recipient);
}
```

Appendix C

Detecting Occlusion

The following sections describe a particular need initially considered when studying Multiple Hypothesis Tracking (MHT) implementation options, across the consulted literature it is mentioned repeatedly the computational and scalability cost of an MHT approach, it is questioned then if it is worth to base the whole tracking in a single MHT flow, or if it could be beneficial to utilize an MHT approach only in some frames, selected based on the probability of an occlusion event to occur in the framed scene, but use a lightweight algorithm on scenes with low probability of an occlusion event, transforming a monolithic tracker as ACE into a flexible dynamic tracker. The trend in newer platforms is to integrate multiple tools that complement each other into providing a clean and accurate tracking result, as seen in [7], [48].

Two main approaches are considered in order to identify scenes with occlusion events, the first one is to check for missing labels or inconsistent nodes in the constructed multipartite graph of a frames sequence (see 2.3). This method, considering that all the blobs are actually soccer players, allows to identify the frames where the tracking algorithm is failing, thus highlighting were it needs some help from a different algorithm. However, this method requires an analytical step, usually performed by a human with knowledge and understanding of the tracking algorithm and failure modes, but also requires the multipartite graph to be available, which implies that the blobs have already been labeled and tracked. It will require an automatic semantic interpreter of the multipartite graphs to clearly classify which failures are due to occlusion and which due to other causes, and just after that is done an occlusion solver may make sense. It is not the intention of this research to develop a programmed module for the semantic interpretation of failure modes during a tracking process, nor to wait until the failure has occurred to fix it, but to be able to detect an scene with high probability of occlusion (high probability of the tracking algorithm to fail due to occluding players) before the players are even identified, but already detected.

The second approach discards knowing which player is which and just requires the blob's positions. This consideration will allow to classify scenes according to how probable it is for an occlusion event

to cause a tracking failure by the agglomeration of blobs in a scene, which can be solved by one algorithm or another, selected based on the scene classification from the objects agglomeration, as some algorithms are best suited for some situations than others [49]; an occlusion solver would be a module that can be enabled/disabled automatically during a tracking process to improve the results on a selected sequence of frames based on the occlusion probability from the objects' agglomeration.

Soccer Player's Spatial Distribution

An occlusion happens when the objects of interest are close to each other from the camera point of view, be it partial of complete occlusion. Using this hint, it is expected that a scene with high probability of occlusion would be one in which soccer players appear to be too close to each other, i.e. an agglomerated image frame. A measure of this agglomeration is required in order to identify an image frame as having high complexity or low complexity for the algorithm tracker to solve due to occlusions.

To do so, a quantitative metric that can relate occlusion probability from soccer player's spatial distribution over the field is put to test [59].

Scene Classification by Player Agglomeration

Taking into account soccer players agglomeration, scenes can be classified as having high agglomeration, moderate agglomeration and low agglomeration, being the first one a frame in which many players are closed together, and the latter one where players are spaced, labeled as Bad-Behaved(high agglomeration), Medium-Well-Behaved(medium agglomeration) and Well-Behaved(low agglomeration) scenes (figure C.3). Therefore, it is expected to encounter more occluded players and higher Occlusion Event Complexity (OEC) in a Bad-Behavedscene than in a Medium-Well-Behavedand a Well-Behavedscenes.

There are different ways in which an occlusion can occur, also, it is possible to have many clusters of occluded players distanced from each other, like in figure C.1 with at least 5 clusters identified inside the soccer field, the current classification still needs more refinement in order to better reflect the possible scenarios, as well as the definition of the OEC.

Figure C.1: Identification of occlusion clusters in a Bad-Behavedscene frame.

Players Dispersion Level measurement

According to [15], [16], [28], there exist three main metrics to study the soccer player's dispersion over the field, Stretch Index, Surface Area, and Team Length. Team Length requires knowing to which team each player belongs, as it is used to describe a team's tactics and positioning over the field. Surface Area metrics may or may not require team belonging information, it is used to describe areas where players step the more into, this to reflect concurrent field zones for different key plays and with team belonging data it can show how well a team executes a maneuver by showing where the players moved. Stretch Index compares the average distance of a team's player and geometrical center, being a measure of how compact or stretched is the team over the field. Although these metrics are commonly used over tracked or identified players and teams (after the final and manually cleaned tracking results are obtained), evaluating over their base idea can be useful while using blobs, as the main piece of information used are the objects position in the field.

To represent the players dispersion over the field, two relative distances are considered: the distance between players and the distance from a global reference point. There may be distanced clusters of occluded players over the field, the global reference can depict this case. There may be an agglomeration of many players in a close area (like in a corner kick) but the players are spaced enough to not occlude, the distance between players depicts this case. From the previous mentioned metrics, the Stretch Index is the one which expression matches best the above needs.

Taking $P_{axis,n}$ as the n'th player's position over $axis = \{X, Y\}$ in frame k, $GC_{axis}(k)$ the position of the geometrical center of the team in frame k, with N the total number of team players in frame k, Stretch index (SI) can be calculated by equation C.1:

$$SI = \frac{\sum_n^N \sqrt{(P_{x,n}(k) - GC_x(k))^2 + (P_{y,n}(k) - GC_y(k))^2}}{N} \tag{C.1}$$

The previous expression has the limitation that it requires to have each team's players identified (and tracking cleaned) in order to obtained $GC_{axis}(k)$. However, if it is ignored that there exist three main cluster of players (two teams and referees), but a whole big cluster of blobs to track, and the geometrical center $GC_{axis}(k)$ is seen as the center of that cluster of objects, SI is, therefore, the magnitude of the average of the distances blob-to-cluster, which describes the first distance desired for the metric to design, the distance between players. If $GC_{axis}(k)$ is replaced by a general reference point, equation C.1 can be rewritten to take the reference's position as a parameter R:

$$SI(R) = \frac{\sum_n^N \sqrt{(P_{x,n}(k) - R_x(k))^2 + (P_{y,n}(k) - R_y(k))^2}}{N} \tag{C.2}$$

Furthermore, if the general reference R does not move throughout the match recording, like the four corners of the soccer field delimiters, it becomes a merely constant that can be obtained from the early stages of the tracking process, in the case of panoramic recordings of the whole field where the four corners are visible and with constant camera perspective.

$$SI(R) = \frac{\sum_n^N \sqrt{(P_{x,n}(k) - R_x)^2 + (P_{y,n}(k) - R_y)^2}}{N} \tag{C.3}$$

To comply with the desired convention, blobs are to be referenced as *players* even though a blob has not yet been identified as one, thus, distances are referred as player-to-cluster, player-to-reference, player-to-player instead of blob-to-cluster- blob-to-reference and blob-to-blob.

Figure C.2 depicts what $SI(R)$ represents, where the red lines are the vectors player-to-reference, and the blue line is the average vector, where $SI(R)$ is the blue line's magnitude. This measure represents the position in the field where the most players are agglomerated.

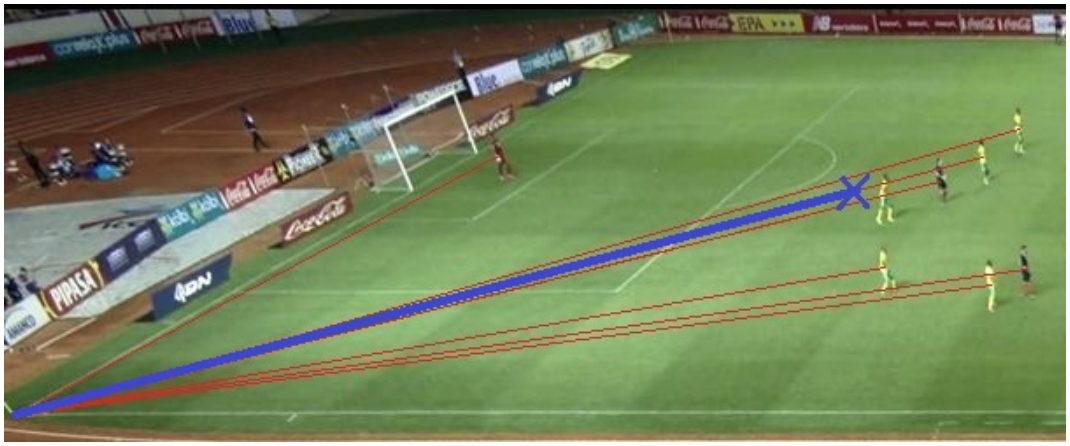


Figure C.2: Graphical description of the modified Stretch Index with a general reference SI(R), using the corner of the field as the reference. Red lines are the vectors player-to-reference and blue line is the average vector. SI(R) is the magnitude of the average vector.

Having identified the average vector, i.e. the place with more agglomeration, the next step is to convert its magnitude into a metric of how agglomerated are the players around that point. To accomplish this, the standard deviation is considered, using the equation C.4, where $PR_n(k) = P_n(k) - R$ is the vector player-to-reference, with $P_n(k)$ the player's position taken from the origin of the image's frame (pixels 0,0), R the general reference from the same origin, $|PR_n(k)|$ the magnitude of the vector player-to-reference, and μ the mean of the vector's magnitudes.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n}^{N}(|PR_n(k)| - \mu)^2} \tag{C.4}$$

Where $SI(R)$ can replace μ, thus the standard deviation σ of the player-to-reference vectors' magnitude around the average vector magnitude is described by equation C.5.

$$\sigma = \sqrt{\frac{1}{N}\sum_{n}^{N}(|PR_n(k)| - SI(R))^2} \tag{C.5}$$

If σ is low, then the players are gathered close around the agglomeration point, if it is high, the players are distant from the agglomeration point. In this sense, a low σ would mean a high occlusion probability due to the players being close to the same point.

Players position affect the shape of the cluster, depending on the reference taken into account, the average vector is dependent on its reference, to include this information it is considered an average of the σ using each corner of the field, finally defining the Players Dispersion Level (PDL) metric,

described by equation C.6, where $c = 1, 2, 3, 4$ is the corner used as reference and R_c is the corner's position.

$$PDL = \frac{\sum_c^4 \sqrt{\frac{1}{N}\sum_n^N (|PR_n(k)| - SI(R_c))^2}}{4} \tag{C.6}$$

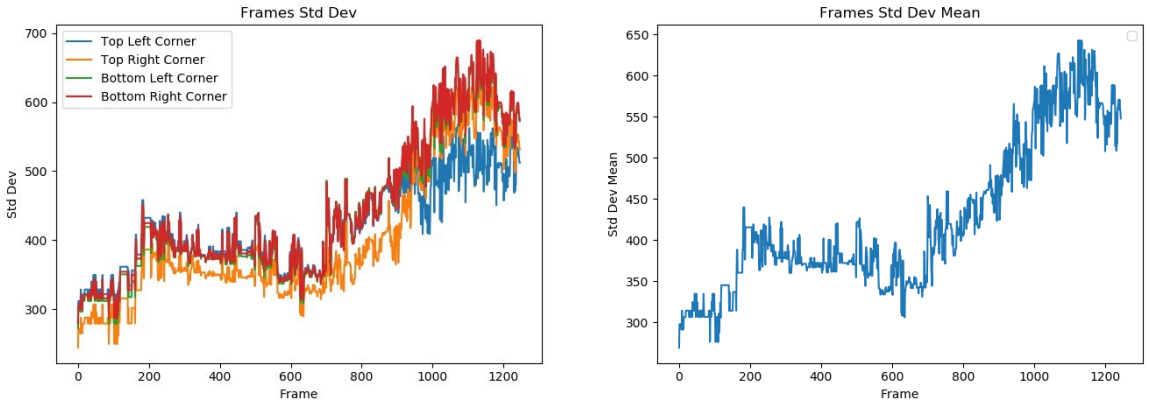
Testing the metric with a short video sequence of 40 s, three main value regions can be extracted that match with the required scenes classification, were a Bad-Behavedhas a low PDL and Well-Behavedhas a high PDL, as seen in figure C.3, for scene classification, and figure C.4 for the standard deviations with each corner as reference and the final PDL calculation for each video frame.

(a) Well-BehavedScene - PDL > 500

(b) Medium-Well-BehavedScene - PDL between 400 and 500

(c) Bad-BehavedScene - PDL < 400

Figure C.3: Soccer player match scenes classification by player's agglomeration



(a) Standard deviation for each corner reference **(b)** Average of standard deviations

Figure C.4: Players Dispersion Level of a 40 s clip from a soccer match using the four corners of the soccer field as references

Accurate thresholds to define each class are still pending, as it requires longer recordings and different matches. The bigger limitation noticed so far is that this metric takes all the blobs as a whole single cluster represented by the average vector, where multiple smaller clusters of occluded players may remain hidden if well spaced from each cluster. Also, as the metric is designed to use blob's positions disregarding player identification stage, there may exist spurious blobs that affect the metrics by adding false players' positions into account to the calculation. Also, this metric is a single value that encompasses the whole scene rather than an actual distribution of the players' agglomeration over the field, keeping its core intention from the Stretch Index.

Kernel Density Estimation for Spatial Distribution Function

Another approach on describing soccer players' spatial distribution is to define an actual distribution function with which to identify field regions with the highest probability of players being agglomerated, thus being a direct metric of occlusion probability.

Soccer players motion complexity has proven to require tracking algorithms designed for non-Gaussian non-linear problems [20], [58] in order to improve the results, as predicting where a player may move is difficult even knowing the team's strategic plan, although it is more probable that the player will continue forward following its inertia, humans can quickly change direction and velocity, as well as drifting on the grass. Thus, a well fitted motion model is not yet developed and rather represented as a complex net of possible actions a player can take in different situations, specially for soccer simulations where virtual assets are expected to behave as a real player, studying teams

strategy instead [5], [76]. Some other studies try to fit goals, team strategy, passes, attack/defense and other in-game actions to different probabilistic functions [52], [87], however, these studies are focused on analysing game strategy and statistics, how players interact and react, and what differentiate elite teams and players from the rest, using clean tracking results. Main metrics to describe players distribution over the field remains Stretch Index and Surface Area [16].

Kernel Density Estimation is a non-parametric statistical tool that allows to estimate the probability density function of a random variable, therefore, having the series of n observations $x_1, x_2, ..., x_n$ taken from the population X with an unknown probability distribution function $f(x)$, the kernel estimate $\hat{f}(t)$ of original $f(x)$, with a kernel function $K(x_i, t)$, is defined by the equation C.7, following convention and implementation details as shown in [95].

$$\hat{f}(t) = \frac{1}{n} \sum_{i=1}^{n} K(x_i, t) \tag{C.7}$$

Considering the current scenario with independent populations X, Y the players positions in the frame k, with pair observations $x = x_1, x_2, ..., x_n$ and $y = y_1, y_2, ..., y_n$, bivariate kernel estimation $\hat{f}(x, y)$ can be described by equation C.8, where kernel function $K(x_i, t) = \frac{1}{h} K(\frac{x-t}{h})$, due to its symmetry property [95], with h the smoothing parameter.

$$\hat{f}(x, y) = \frac{1}{n h_x h_y} \sum_{i=1}^{n} K(\frac{x_i - x}{h_x}, \frac{y_i - y}{h_y}) \tag{C.8}$$

Using blobs positions from a 40 s clip of a soccer match and the Python library Seaborn [94], with a Gaussian kernel function and smoothing parameter calculated by the Seaborn's function *kdeplot* internal calculations, it is possible to generate a 2D plot of the bivariate estimated probabilistic function for the players' positions distribution over the field for each frame of the clip (see figure C.5).

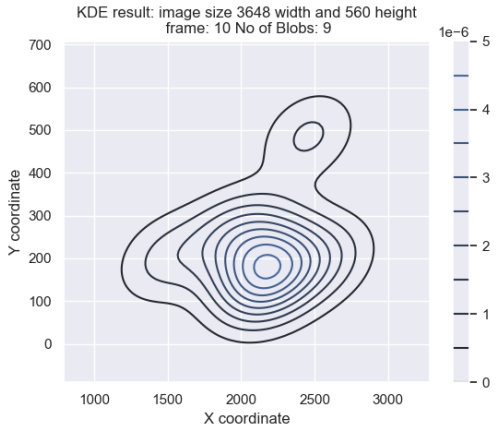
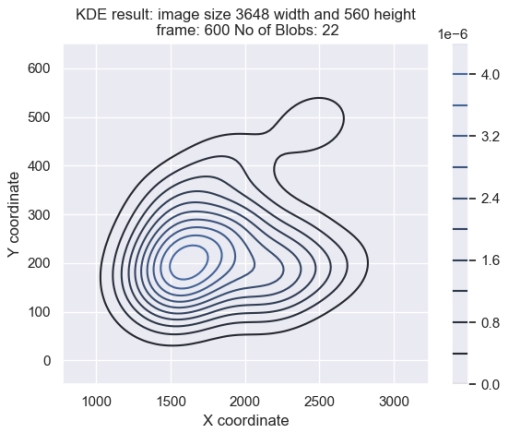


Figure C.5: Estimated bivariate probabilistic function of soccer players spatial distribution for frame $k = 10$

Comparing results with PDL, it can be seen that they coincide in which scenes players are more likely agglomerated and in which they are spaced. Fro frame $k = 600$ the $PDL < 400$, which means it's basically a Bad-Behavedscene, which can be seen in the KDE plot by a stretched shape (figure C.6), while the frame $k = 1100$ with a $PDL > 500$ has a more elongated shape with lower peaks, being a Well-Behavedscene (figure C.7).

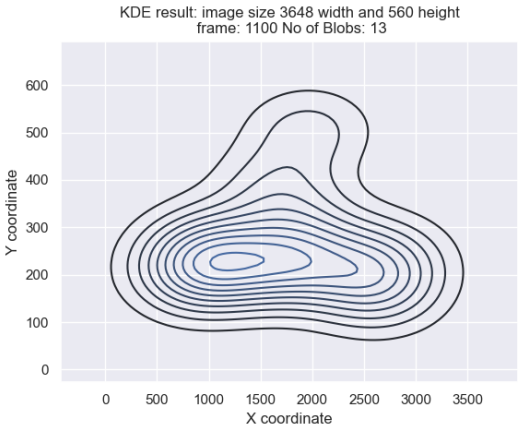
This method presents an advantage over the PDL one as it allows to identify probabilistic regions instead of a global value, thus it highlights clustered areas, being useful to classify scenes for its general agglomeration density and also to distinguish image regions of high agglomeration, allowing to detect independent clusters like the one located at (x=2500, y=500) in figures C.6a and C.7a, providing a tool to not only select frames to use the occlusion solver on, but also to select image regions for a more localized and thorough occlusion solver, if desired.



(a) Estimated bivariate probabilistic function

(b) Respective frame from soccer match clip

Figure C.6: Estimated bivariate probabilistic function of soccer players spatial distribution for frame $k = 600$ and $PDL < 400$, classified as Bad-Behavedscene



(a) Estimated bivariate probabilistic function

(b) Respective frame from soccer match clip

Figure C.7: Estimated bivariate probabilistic function of soccer players spatial distribution for frame $k = 1100$ and $PDL > 500$, classified as Well-Behavedscene

Occlusion Probability and Window Bending

Based on the discussion from C and 4.4, it can be useful to have the flexibility to dynamically modify the size of the batch to be processed, firstly to ensure an occlusion event is fully contained in the same batch and thus the same Multipartite Graph (MPG), secondly to be able to select a bunch of frames to be processed with by a lightweight algorithm or a heavy duty one depending on the scene complexity.

Two metrics to determine when to extend the batch and when to cut it are explored (see C), the core concept is the same for both, use the player blob agglomeration to associate the probability of an occlusion event to occur and cause problems to the tracker; the more agglomerated the players the more occlusion events and the more complex the occlusions are. The first considered metric is the Kernel Density Estimation (KDE) (see C), there are some free libraries that implement the KDE, one in particular is quite easy to use ([94]), but its use as part of ACE platform was discarded as it is a Python library and some modifications were required to adapt ACE data structures to the required input parameters of the library functions. A custom implementation of KDE was developed for the particular needs of ACE platform. Even though the code is available as part of the tracker, the use of this metric was discarded due to the considerable increase in time execution only to generate one KDE

representing one frame of video. ACE platform operates with a panoramic image result of stitching two images from 4K cameras each with UHD of 3840x2160 px, generating images of 16 588 800 px. The KDE generates one value for each pixel, which is then interpreted as a heat or relief map that matches 1:1 the image. For each pixel, the density estimate is the result of the kernel function applied to the distance between the pixel and each blob position (22 players + 3 umpires), which results in calculating the kernel function $25 * 16588800 = 414720000$ times, then generate the bivariate density map for a single frame, if we consider a full soccer match of 90 min recorded at 30 FPS, the kernel function is to be calculated 67 184 640 000 000 times just to get a hint on how to split or extend the MPG to apply a tracking algorithm. This costly function is reserved for future research where integrating KDE is at the center of the research and not as a suplementary tool.

The second metric consist on the PDL, a more lightweight metric that can be directly used as a threshold to classify scenes (see C). Three scene classes are considered: well-behaved, medium well-behaved and bad-behaved scenes. If a batch contains a bad-behaved scene, the batch size is expanded until a medium well-behaved or a well-behaved scene is found, so that the new batch of frames to process fully encloses a scene with high probability of complex occlusion, this process is named 'window bending' as we are bending the batch window size to fit the required scene.

The thresholds to classify the scenes are determined from the highest expected PDL value considering only the expected players are detected, since this metric takes as input the detected blobs, spurious blobs can alter its value by increasing the apparent agglomeration of blobs in the scene. The highest PDL corresponds to the highest possible dispersion of players across the field, a more fitting model could have been implemented, but for simplicity an arbitrary distribution of players is considered, the placement is distributed along the field considering the dimensions based on the image coordinate system.

$nx \leftarrow 8 \times 2$

$ny \leftarrow 3 \times 2$

$i, j \leftarrow 0$

$xl, xr \leftarrow 0$

$delta_y, delta_x \leftarrow 0$

$y, x \leftarrow 0$

$points \leftarrow$ empty list

for $j \leftarrow 1$ **to** ny **with step** 2 **do**

$delta_y \leftarrow \frac{bottomLeftCorner.y - topLeftCorner.y}{ny}$

$y \leftarrow \text{int}(delta_y \times j + topLeftCorner.y)$

$xl \leftarrow topLeftCorner.x + (y - topLeftCorner.y) \times \frac{bottomLeftCorner.x - topLeftCorner.x}{bottomLeftCorner.y - topLeftCorner.y}$

$xr \leftarrow topLeftCorner.x + (y - topRightCorner.y) \times \frac{bottomRightCorner.x - topRightCorner.x}{bottomRightCorner.y - topRightCorner.y}$

$delta_x \leftarrow \frac{xr - xl}{nx}$

for $i \leftarrow 1$ **to** nx **with step** 2 **do**

$x \leftarrow delta_x \times i + xl$

$points$.push_back($cv :: Point2f(x, y)$)

end for

end for

After calculating the highest PDL, the range if divided in thirds to define the thresholds of the three categories, where the lower the PDL the greater the agglomeration and the probability of occlusion.

Similarly with KDE, the use of this metric and window bending is discarded from the final implementation of the ACE tracker, as it was noticed that both Multiple Multipartite Graphs (MMPG) and Track Oriented Multiple Hypothesis Tracking (TOMHT) did not considerably increase the tracking execution time. This implementation, however, remains a tool for future developments.