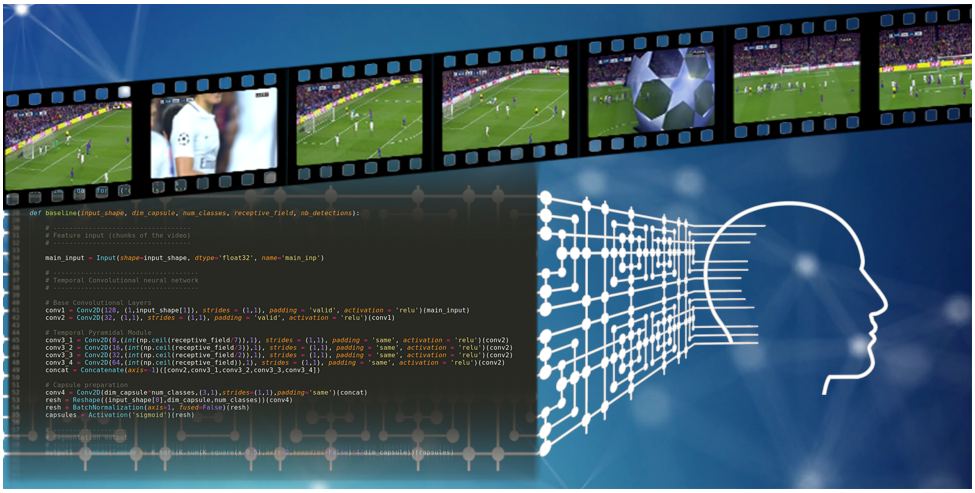
Montefiore Institute
Department of Electrical Engineering
and Computer Science
Faculty of Applied Science
University of Liège

REAL-TIME SEMANTICS IN VIDEO SEQUENCES

Anthony CIOPPA

PhD dissertation



Marc	VAN DROOGENBROECK	University of Liège (Belgium)	*Supervisor*
Louis	WEHENKEL	University of Liège (Belgium)	*President*
Adrien	DELIÈGE	University of Liège (Belgium)	*Jury Member*
Gilles	LOUPPE	University of Liège (Belgium)	*Jury Member*
Olivier	BARNICH	EVS (Belgium)	*Jury Member*
Silvio	GIANCOLA	KAUST University (Saudi Arabia)	*Jury Member*
Thomas B.	MOESLUND	Aalborg University (Denmark)	*Jury Member*

December 2020

REAL-TIME SEMANTICS IN VIDEO SEQUENCES

Anthony CIOPPA

Montefiore Institute
Department of Electrical Engineering and Computer Science
Faculty of Applied Science
University of Liège

December 2020

Abstract

The study of semantics intends to provide meaning to data. In the case of video sequences, semantics allows to derive an analysis of the scene, on which many real-world applications can rely. To this extent, we start by defining two levels of semantics that can be extracted from videos. First, we define as *low-level semantics* every information describing the natural content of the video, comprising the objects and the environment of the scene. Second, *high-level semantics* characterizes the interpretation of the events occurring in the scene, which relates to a deeper understanding of the role of the elements composing this scene.

In the first part, we explore several approaches to extract low-level semantics from video sequences in real time, as most current state-of-the-art methods are rather slow. In particular, we focus on three types of low-level semantics: motion detection, semantic segmentation and object detection. As a first contribution, we develop an asynchronous combination method to leverage the output of a slow segmentation network to improve the performances of a real-time background subtraction algorithm, while keeping real-time inference. As a second work, we present a novel method to train a fast segmentation network by leveraging the output of another slow, but performant, segmentation network while constantly adapting to the latest video conditions. Then, we show that this method, called *online knowledge distillation*, also proves to be effective for detecting players on a soccer field, even when the two networks process videos with different modalities and fields of view.

In the second part, we focus on high-level semantics describing the events taking place during a soccer game. First, we leverage low-level semantics to progressively produce a higher-level understanding of the game and present a simple, yet effective, semantic-based decision tree to segment the following game phases: goal or goal opportunity, attack, middle and defense. In a second approach, we develop a novel network architecture coupled with a context-aware loss function to spot game events such as goals, card and substitution, and show that it achieves state-of-the-art performances on the SoccerNet dataset. As a final contribution, we publicly release a novel dataset containing high-level semantic annotations, comprising a complete set of game events and semantics related to the editing of the TV broadcast. This allows us to define four challenging tasks: action spotting, camera shot temporal segmentation, camera shot boundary detection, and replay grounding. We hope that this dataset will become the reference for high-level semantics in soccer videos.

Résumé

L'étude de la sémantique permet d'interpréter des données. Dans le cas de séquences vidéos, cette sémantique fournit une analyse de la scène qui est utile dans de multiples applications pratiques. Afin de formaliser cette notion, nous définissons deux niveaux de sémantique pouvant être extraite des vidéos. Dans un premier temps, nous définissons la sémantique de bas niveau comme toute information décrivant le contenu naturel de la vidéo, c'est-à-dire les objets et l'environnement constituant la scène. Dans un second temps, nous caractérisons la sémantique de haut niveau comme étant l'interprétation des événements survenant dans cette scène.

Dans la première partie de cette thèse, nous explorons de nouvelles approches permettant d'extraire de la sémantique de bas niveau dans des séquences vidéos en temps réel étant donné que la plupart des méthodes actuelles sont trop lentes. Nous nous focalisons en particulier sur trois catégories de sémantique de bas niveau : la détection de mouvement, la segmentation sémantique et la détection d'objets. Notre première contribution consiste en une méthode de combinaison asynchrone pour améliorer les algorithmes de détection de mouvement via l'introduction d'informations sémantiques provenant d'un algorithme de segmentation tout en conservant une approche en temps réel. Ensuite, nous présentons une nouvelle méthode d'entraînement de réseaux de neurones supervisée dans laquelle un réseau de segmentation rapide est entraîné tout au long de la vidéo grâce à la sortie d'un second réseau plus lent, mais plus précis. Cet entraînement, que l'on appelle la distillation en ligne, permet au réseau de s'adapter aux dernières conditions de la vidéo et d'améliorer ses performances tout en restant en temps réel. Finalement, nous montrons que cette méthode d'apprentissage est également adaptée à la détection de personnes dans le cas où les deux réseaux traitent des vidéos avec des modalités et des points de vues différents de la même scène.

Dans la seconde partie, nous nous focalisons sur la sémantique de haut niveau, et plus particulièrement l'interprétation des événements dans des matchs de football. Dans un premier temps, nous montrons qu'il est possible d'utiliser l'information sémantique de bas niveau pour construire progressivement une compréhension de plus haut niveau du jeu. Nous proposons une méthode basée sur un arbre de décision sémantique pour segmenter les différentes phases de jeu : goals ou opportunités, attaque, défense et jeu médian. Nous proposons également une seconde approche pour détecter les événements de jeu tels que les goals, cartes et substitutions basée sur une nouvelle fonction de coût prenant en compte le contexte temporel entourant ces événements. Grâce à notre méthode, nous établissons un nouvel état de l'art sur la base de données Soccer-

Net. Comme dernière contribution, nous publions une nouvelle base de données autour de la sémantique de haut niveau. Cette base de données comprend des annotations pour l'ensemble des événements de jeu ainsi que liées à la production du flux télévisuel. Grâce à ces nouvelles annotations, nous définissons quatre tâches : la détection d'événements, la segmentation temporelle du type de caméra, la détection des changements de caméras et une dernière tâche qui vise à lier chaque rediffusion d'une action avec le moment durant lequel elle s'est déroulée. Nous espérons que cette base de données deviendra la référence en terme de sémantique de haut niveau dans des séquences de football.

Acknowledgments

First and foremost, I would like to thank my supervisor Marc Van Droogenbroeck for coaching me during my PhD. He supported this project and believed in me to reach our goals. He consistently helped me get through the difficulties along the way and I strongly believe that his guidance really made what this thesis has become today. Besides, our wonderful conference trips will be printed indelibly in my memory, as well as our long working sessions before a deadline, such as that one time at 2am near Yellowstone National Park. In a nutshell, I thank him for providing me with such an amazing doctoral experience.

One of the most memorable period of my PhD surely corresponds to my three months research stay at Aalborg University in Denmark. Therefore, I would like to express my gratitude to Thomas B. Moeslund and Rikke Gade for such a warm welcome and their excellent guidance throughout the two research projects that we conducted. I would also like to thank Noor Ul Huda, my co-author, Chris Holmberg Bahsen for his guidance, Joakim Bruslund Haurum, Malte Pedersen and the rest of the lab for their great scientific presentations and their always amusing "drink and play" afterwork.

Let's not forget that doing a PhD costs a lot of money. Therefore, I would like to thank the FNRS for funding my research and travels via their FRIA grant. I would also like to express my gratitude to the Research and Technologies Department of Wallonia, Belgium, for their financial support on the hardware and the hiring of the student annotators.

When I was not outside of Belgium, I could rely on the best laboratory teammates. We built such an amazing friendship that coming to the office did not feel like work. I hope we can keep our daily "Rikiki" ritual once this covid period is behind us. Therefore, I would like to profoundly thank Anaïs Halin, my always joyful office roommate, Sebastien Piérard, Marc Braham, Michael Fonder, Antonio Sutera, Damien Gérard, Nicolas Vecoven and all my other colleagues that made this PhD so much enjoyable on a daily basis. I would also like to thank Silvio Giancola for his amazing guidance through the CVPR submissions.

At this point, the attentive reader most probably think that I forgot the key player in my team composition. Well, that is because I wanted to express my *ultimate* gratitude to Adrien Deliège. Not only is he my co-first-author on most of ~~my~~ our work, but he is also my YouTube co-host, my travel partner, my pool rival, my cooking student, my scientific mentor and most importantly my dear friend. I will always be grateful to him for what he has brought me in research, but also for his friendship. I sincerely hope that we can keep

on collaborating on many projects in the future.

Another special thanks goes to my beloved friends Raphael La Rocca, Maxim Henry, Maurice Genet and Brice Lumia for their unconditional support throughout this thesis and my whole life in general. I would also like to thank my two-year roommate Tom Michel who thought me all I know about soccer and with whom I learned the basis of computer vision and deep learning. Finally, I would like to thank my other awesome friends Colin Palmaerts, Christophe Blom-Peeters, Axel Vijgen, Charlotte Schöpges, Laurent Portelange, Alex D'heur, Christophe Loix, and my two childhood friends Cyrille and Jérôme Francisco.

Finally my deepest thanks go to my family. Especially, I would like to thank my mother, Sabina Ceccato, and my father Carmine Cioppa for their unconditional love. They always supported my projects and helped me get to my objectives, no matter the cost. There is no word to express the gratitude that I have for them. I would also like to thank my sister, Kelly Cioppa, for our amazing *siblingship* and I wish her good luck for her own PhD, no pressure. Also, I would like to thank my grand-parents with whom I have wonderful childhood memories.

Finally, as a cherry on the cake, I would like to profoundly thank my 7-year best friend and lover Céline Janssen for her love and support throughout all these years. She is the one that took care of me during the long nights before the deadlines and that made sure that I received enough food and drinks to keep my brain going. I am sure that I wouldn't have made it so far in this journey without her.

Contents

CHAPTER 1

Introduction

Contents

1.1 Motivation and objectives

Semantics has been studied in many fields across science, literature and philosophy. Even though its exact definition may vary depending on the field, a common feature across all definitions is the notion of "meaning". For this reason, semantics can be considered as information that is, somehow, meaningful for a system [114]. This definition is not complete and leaves some grey areas as what should be considered as meaningful information. In this thesis, we restrict the scope of semantics and only consider semantics extracted from single frames (intra-frame semantics) or video sequences (inter-frame semantics). More specifically, we focus on soccer video content and aim at extracting the same set of semantics that a human viewer would be able to retrieve when watching the game. As an illustration, let us take the example of a soccer video. As humans, our brain is able to extract meaningful information when watching the video, *e.g.*, we are able to see and maybe recognize the players that are present on the field, where they are and what action they might be doing. This kind of information that our brain extracts about the content of the video can be designated as *semantics*. By contrast, it is not straightforward for a computer-based artificial intelligence to retrieve this semantics. The challenge tackled in this thesis is thus to develop automatic computer-based methods to extract different types of semantics in videos. In the same manner, we also consider that semantics can be extracted from other types of semantics, *e.g.*, from the information that "all players are running towards one goal", semantics about the game may be inferred, like "their is a goal opportunity". To summarize, we define the notion of semantics in this thesis has follows:

DEFINITION OF SEMANTICS

Information that has a meaning, in the broadest sense of the term, for a human observer and that can be extracted at different levels.

This semantic interpretation of sports game is a challenging topic of research in the domain of computer-based techniques [183] and is still, widely unsolved. In this context, we developed several methods to extract semantics at different levels from the video which gets us closer to this complete understanding of soccer games and videos in general. Before getting into the details, it is important to define the different levels of semantics that we intend to derive (Section 1.1.1), the challenges of the current deep learning approaches aiming to extract them (Section 1.1.2), and the potential application of semantics to solve real-world problems (Section 1.1.3).

1.1.1 Low vs high-level semantics

When watching a soccer game, it is clear that any human viewer is able to get at least some semantics about what is shown. For example, it is straightforward for him to retrieve the position and pose of the players or know where the field and the public are in the image even if the viewer has no experience in watching soccer. Additionally, if the viewer also has some knowledge about soccer, he will be able to understand the events occurring during a game, such as an unsanctioned foul or an offside, or even understand the different tactics of each team. Through this example, it can be inferred that semantics is present at two different levels of understanding. The first one relates to the description of the scene and requires no particular knowledge about what is happening in the scene, while the second level is linked to the interpretation of the scene, which requires some insights about the events that occur in the scene.

Therefore, we define as **low-level semantics**, information that relates to the natural content of the scene. This level of semantics describes the objects of the scene as well as the environment in which the scene takes place. In the particular case of soccer videos, this can refer to detecting or segmenting the players, determining which players are in motion, recognizing the players, or segmenting the field, the field lines and the public. To provide these descriptors of the scene, low-level methods mainly use the raw information contained in the video to extract local semantics in the frames of the videos, but do not infer any interpretation from these descriptors.

In complement, we define **high-level semantics** as information that relates to the interpretation of the scene. This level of semantics describes what is happening in the scene. In soccer games, this can relate to the actions that occur during a game, such as goals, cards or substitutions. Furthermore, high-level semantics may also refer to the way the scene is shown to the viewer. For example, why a certain type of camera is shown on television by the broadcast producer is also linked to the interpretation of the game,

as this choice unveils its intention to convey particular emotions. In order to provide this level of interpretation, high-level methods may be based either on the raw information or on low-level semantics.

Even though these definitions seem to mutually exclude each other, it can still be tricky to classify certain type of semantics in a single level, because this level can depend on its use in the final application. For example, information about which camera is shown can be both seen as a description of the environment, *e.g.*, the close-up camera is shown, or as an interpretation of the scene, *e.g.*, when the close-up camera is shown; this suggests that an important event might have happened involving the shown player. In this thesis, we mostly describe semantics as low or high depending on its use in the considered work only. An illustration of these levels of semantics in the case of soccer can be found in Figure 1.1.

1.1.2 Challenges of deep learning approaches

In recent years, supervised deep learning approaches have shown impressive performances on most vision-based problems such as semantic segmentation object detection or video understanding. Even though they have completely beaten unsupervised methods on most evaluation datasets, deep learning approaches are also known to have some major drawbacks, especially in real-world or industrial applications. We found that the following three challenges are the most critical:

1. **Annotation**. Most supervised approaches require huge datasets containing ground-truth annotations that often have to be provided manually. This annotation process is time and money-consuming and has to be handled for every new specific task or application. In the case of sports videos, there are few large datasets, and the ones that exist are either specific to a single sport or are privately owned by companies. Therefore, it is quite arduous for academic researchers to develop a large variety of methods, since they often do not have the time or the funds to annotate a new dataset for every new research paper. Even if they have the funds to annotate the data, they often keep it private since it gives them a competitive advantage over other researchers or industries. These private dataset are a problem as well since reviewers are not able to reproduce the results of the proposed methods to ensure that its reported performances are valid. One way to alleviate the annotation issue is to use unsupervised approaches, but they have shown much lower performances than their supervised counterparts. In this thesis, we focus on replacing the need for manual annotation when possible using novel strategies to efficiently train supervised networks. We also propose a novel large dataset for video understanding in soccer that we publicly release for the scientific community.

2. **Speed vs performance**. It has been observed in many different tasks and chal-

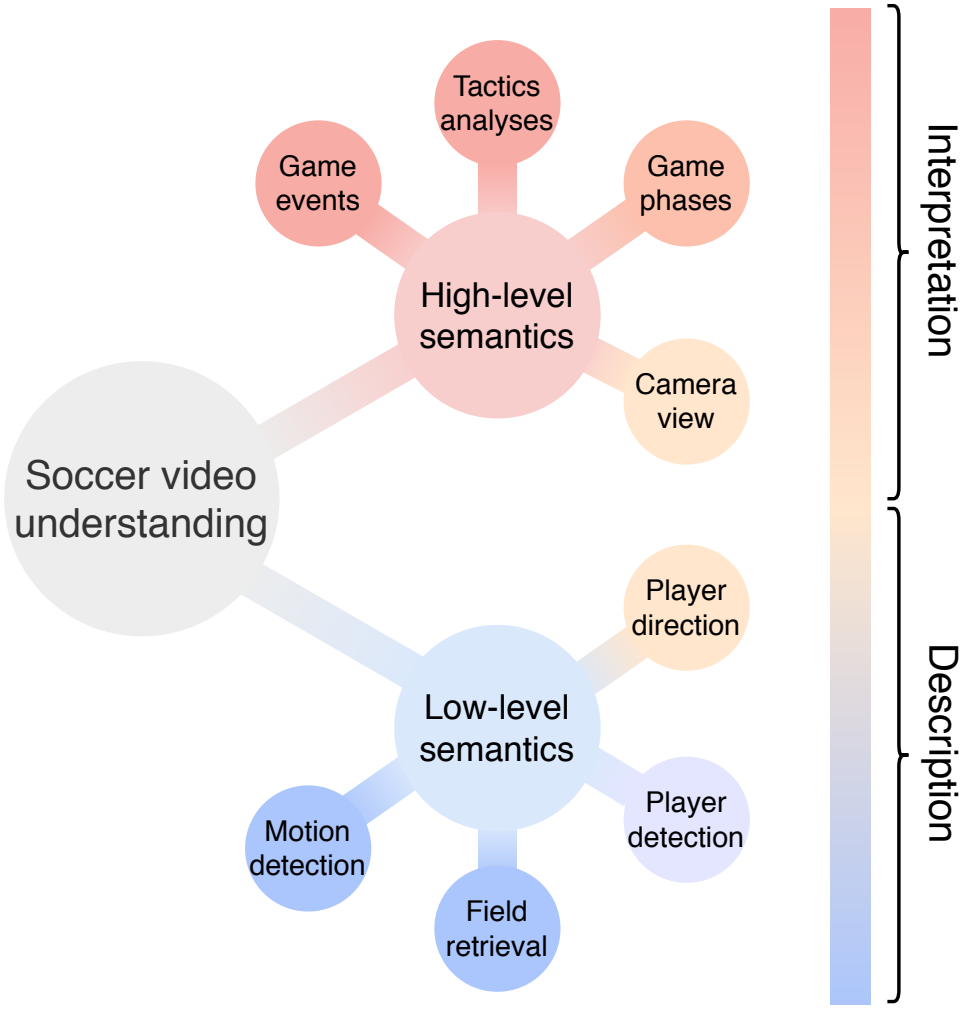


Figure 1.1: Levels of semantics. Illustration of low-level and high-level semantics in soccer videos. Low-level semantics describes the content of the scene while high-level semantics corresponds to the interpretation of the soccer game. For certain types of semantics, it can be difficult to assign it to a particular level of semantic since it depends on its use. Therefore, the levels of semantics are illustrated as a continuum among two separated ensembles.

lenges that the state-of-the-art methods are often very slow. For instance, on one of the most famous dataset for semantic segmentation, called Cityscapes dataset [39], the current best algorithms [32, 221] are rather slow, while the real-time ones [143, 203, 210, 219] show much lower performances. Given that this dataset is meant to serve the autonomous vehicles industry, it is essential that both performance and speed-based criteria are met simultaneously, which is not the case at the moment. These two aspects can also be required in sports video understanding to provide real-time accurate information about the ongoing match. The methods developed in this thesis thus focus on real-time performances, especially the ones that extract low-level semantics since they are often used as building blocks for higher-level semantics in various applications. Therefore, our low-level methods propose new strategies to obtain state-of-the-art performances while keeping the real-time constraint.

3. **Scene-specific vs universal training**. A last challenge to consider is the generalization capability of these supervised models on unseen data. In fact, overfitting the training on a particular dataset can lead to poor performances on data out of the original distribution. To overcome this issue, some dataset propose input data covering the widest distribution possible. This leads to the development of universal methods that often have to be computationally heavy to encode all these possible scenarios. This might be an overkill in scene-specific applications, such as soccer, where much lighter methods could be sufficient for the required tasks. For example, there is no need for a soccer-centric method to be able to detect planes or boats, as they do not show up in the scene. Focusing only on relevant classes has the advantage reducing the computation time and memory of such methods, while boosting their performances on the relevant classes. Therefore, we aim at developing scene-specific methods since they are more performant and faster. In this objective, we describe a novel strategy to ensure generalization capabilities on unseen soccer games, by proposing a novel online knowledge distillation method.

1.1.3 Potential applications of semantics

The sports industry has been heavily affected by the global 2020 pandemic, with many games canceled or held behind closed door. Live production companies have thus been put against the wall to produce quality content without any spectators in the stadium, which tends to make the game emotionless. To alleviate this issue, clever solutions were rapidly proposed, such as a virtual replacement of the public in the stadium to fake the presence of supporters. Also, artificially recreated ambient sounds that seamlessly blend with the game events and phases were added to provide emotions to the viewer. The rapid management of this unprecedented crisis was made possible thanks to the recent advances in computer vision and deep learning, on which these technologies rely.

In a more general context, the production of live personalized content is believed to

be the next step in broadcast production and will change the way we watch sports. For example, a casual viewer would be much more immersed into the game if the name of the most famous players were highlighted above their head, while an advanced soccer fans might be more interested in the speed performances of his favorite player during the game, or some highlighting of the defense lines. This is a key financial aspect for production companies and the first one to deliver an attractive and reliable product will most probably take a large proportion of the market shares. In order to achieve these goals, both low and high-level semantics are undeniable assets. This shows that there is a growing need for real-time semantics as live production and the viewers are ready for the next generation of sports content.

Likewise, the automatic production of live content is an emerging business driven by the high production cost of broadcasted games. Semantics about which camera shot is shown or should be used is relevant for an automatic production of a soccer game, or the generation of after-game highlights. The benefit of such automatic methods are numerous, both on the economic and the social sides. Indeed, amateur leagues could benefit from such methods by proposing, for a fraction of the price, the same type of content that only the richest and most famous leagues are able to afford right now.

Following this trend, automatically generated audio comments could replace the commentators for low-budget broadcasts. In fact, some basics of this technology already exist in soccer video games such as FIFA or PES. These games mainly use pre-recorded audio samples to produce realistic comments, where the virtual commentator relies on in-game data about the players and the game events. With the advances in natural language processing and audio generation, it is now conceivable to generate an audio description of computed semantics rather than in-game data and produce some high quality comments alongside the video of an amateur soccer game.

The applications presented above are only few examples among all the possibilities. Even though this thesis does not solve the problem of video understanding, the developed building blocks get us closer to a fully automatic description and interpretation of a soccer game. Finally, real-time semantics such as semantic segmentation is useful in many other fields like autonomous driving to let the car know, with a minimum of delay, where the objects and the road surrounding it are located. In video surveillance, motion detection needs to be accurate and fast in order to trigger an alarm signal when a person enters some restricted area. Since most of these systems are critical and need to be implemented in restricted-resource environments, there is a need for fast and accurate extraction of low-level semantics.

1.2 Thesis outline and original contributions

In the remaining part of this document, some chapters are entirely dedicated to a specific scientific publication. Therefore, these chapters comprise similar content com-

pared to the original publication, including same sentences and figures. Nonetheless, since there are no more restrictions on the length of the document, additional information is included when relevant and the presentation of some methods are modified to fit the theme of this thesis, which revolves around semantics. Also, some links between the chapters are included to ensure an easy reading throughout the document. However, as these chapters relate to different domains of applications, a separate chapter englobing each and every individual related work would be too heavy to read. Therefore, we provide some related works inside of each chapter, specific to the domain of each paper.

The structure of the document itself is separated into two main parts, each covering a specific level of semantics, along an additional appendix part. Part I, comprising Chapters 2, 3 and 4, focuses on low-level semantics, while Part II, comprising Chapters 5, 6 and 7 focuses on high-level semantics. Eventually, a general conclusion summarizing the different works is presented in Chapter 8. For some chapters, we provide additional details in a related appendix in PART III.

This thesis has lead to several original contributions that are individually listed for each technical chapter in this section. It is interesting to note that even though most of the developed methods are applied to soccer videos, they can be generalized to different contexts, *e.g.*, security monitoring, autonomous driving or other sports. Besides, we always aimed at making our work accessible to the scientific community. In this objective, we produced several vulgarization videos on YouTube[1] depicting in a simple way the basics and results of our methods. Finally, we also released some Python open-source codes on GitHub[2] for most of the works described in this thesis. All these resources should allow any researcher to easily reproduce the main results of each of the presented research papers. A short summary of each chapter and their respective contributions can be found hereafter.

Chapter 2 presents a method for producing low-level semantics that describes which pixels belong to objects that are in motion. Specifically, it introduces the concepts of background subtraction and semantic segmentation, which are both used in later chapters. The method that we propose efficiently combines any background subtraction algorithm with the output of a semantic segmentation network. The novelty introduced in this chapter consists in performing this combination asynchronously, *i.e.* independently of their respective frame rates. This allows to improve the performance of any background subtraction algorithm, without increasing its computation time. In the case of a real-time background subtraction algorithm, such as ViBe [12], we show that our method reaches performances close to the ones of non-real-time state-of-the-art algorithms.

Contributions. **(i)** We propose a novel method for the task of background subtraction.

[1] ACAD Research: https://www.youtube.com/channel/UCYkYA7OwnM07Cx78iZ6RHig
[2] Anthony Cioppa's GitHub: https://github.com/cioppaanthony

(ii) We alleviate the problem of the slow computation of semantic segmentation by substituting it for some frames with the help of a newly introduced change detection algorithm. This makes our method usable in real time. **(iii)** We show that at a semantic frame rate corresponding to real-time computations, we achieve top performances, meaning that our substitute for semantics is adequate. **(iv)** We show real-time state-of-the-art performances on the CDNet 2014 dataset.

Chapter 3 proposes a new method to produce real-time *student* networks from a non-real-time *teacher* network through the novel concept of online distillation. This method allows to train the student network during the video so that it can automatically adapt to the latest video conditions. To do so, we use a universal teacher network that produces some very precise ground-truth annotations at a much slower frame rate. This method has three advantages: (1) there is no need for manual annotations to train the student network, (2) the performances of the student quickly reach the ones of the teacher, while being real time, (3) there is a guarantee that the student network will perform well on any new video.

Contributions. **(i)** We propose a novel method for human segmentation during live sports events. **(ii)** For a given match, our method produces an excellent segmentation network that evolves during the match, without having to manually annotate a single frame. **(iii)** We demonstrate the superiority of adaptive scene-specific networks over pre-trained ones. **(iv)** We show that the student may outperform sometimes its teacher in particular situations.

Chapter 4 extends the previous online distillation method in a more general case, where the teacher and the student networks operate on two modalities that have different views of the same scene. The objective consists in detecting all players on an amateur soccer field from a fisheye video stream that covers the whole field. To do so, we use a thermal camera filming the same scene, but with a narrower field of view. On this type of camera, the players can be detected more precisely than on the fisheye one, but the field of view does not cover the whole soccer field. Therefore, the teacher produces some ground truths from the thermal camera that are then transferred to the fisheye camera in order to train the student. Since the thermal camera only represents a small portion of the fisheye camera, a scene-specific data augmentation process is designed to completely cover the field of view of the fisheye camera. Through this work, we show the great potential of online distillation in an apparently complicated setup for a real industrial application.

Contributions. **(i)** We provide a generalization of online distillation in a multi-modal and multi-camera setup. **(ii)** We show how two different image modalities and fields of view can be combined in a student-teacher distillation approach. **(iii)** We show how a student network can be trained to detect players outside the field of view of the teacher,

through a combination of a custom data augmentation process and a motion detection algorithm.

Chapter 5 shows how the low-level semantic information described in Part I can be used to build higher levels of semantics in the case of soccer videos. In particular, it focuses on extracting semantics about the pose of the camera and the position and average movement of the players from low-levels semantics such as a field, player, or field line segmentation masks. This is done using domain knowledge and exclusively computer-vision heuristics. Finally, we show how this type of semantics, we can be used to extract high-level semantics such as game phases by designing an intuitive semantic-based decision tree.

Contribution. **(i)** We show a simple way to leverage low-level semantics to produce high-level semantics in the case of a soccer game. **(ii)** We propose the task of game phase segmentation. **(iii)** We propose an intuitive pipeline based on semantics for temporally segmenting game phases in a soccer game.

Chapter 6 describes a novel deep learning approach for action spotting in soccer videos. The objective is to retrieve all actions occurring during a soccer game from the TV broadcast. We design a novel context-aware loss function that takes into account the temporal context surrounding the action. This loss is used to enforce a temporal semantic segmentation of each class of action to which we append a detection module to precisely define the spots. This allows us to reach state-of-the-art performances on the SoccerNet dataset. Finally, we show that the temporal segmentation and the output of the detection module can be used to automatically generate highlights of the game.

Contributions. **(i)** We present a new loss function for temporal action segmentation and the task of action spotting, which is parameterized based on the context surrounding the actions. **(ii)** We show that our network architecture combined to our loss function improves the previous state-of-the-art performance for the action spotting task of SoccerNet by 12.8% **(iii)** We provide detailed insights into our action spotting performance, and showcase a possible application for the automatic generation of highlights.

Chapter 7 introduces novel high-level semantic annotations for the SoccerNet dataset as its original version is limited, with only 3 classes of actions and the single task of action spotting. First, we raise the number of classes for action spotting from 3 to 17 classes, representing all possible actions in a soccer game. Also, we focus on the spotting of actions that are not shown on TV broadcast, but that the algorithm must still be able to detect based on the surrounding context. Second, we propose some novel semantic annotations that relate to the editing of the TV broadcast, with a particular focus on replays. This allows us to propose novel tasks and applications for the automatic production of video content related to soccer.

Contributions. **(i)** We present a proposal to expand the SoccerNet dataset by extending its action spotting annotations to 17 classes and by providing new annotations related to the editing of the TV broadcast; these additions will be part of the coming new SoccerNet-v2 dataset, **(ii)** We propose an extended action spotting task, based on unshown action spotting, involving the context rather than the visual information. **(iii)** We define several tasks related to the editing of the TV broadcast. **(iv)** We propose a new task that aims at linking the replays to their corresponding live action spot.

Chapter 8 summarizes the main results for low-level and high-level semantics and concludes on the solved tasks of video understanding. It also provides some possible practical applications of our methods and describes further work.

1.3 Publications, patent and awards

This thesis has led to several peer-reviewed publications in top conferences and journals. It is interesting to note that in our research domain *i.e.* computer vision and deep learning, some conferences are as important in terms of peer recognition as some journals. In particular, the CVPR conference places itself in the top-5 of publishers, all categories [4], in front of all other journals of the same domain.

We are also proud that three of our presented works received the best paper award at the CVsports workshops at CVPR 2018, 2019 and 2020. This workshop was particularly appropriate for my thesis as it aims at publishing research in computer vision and deep learning that have a particular focus in sports. As a final contribution, the method presented in Chapter 2 is protected by a US patent. A complete list of my publications can be found hereafter:

Publications in journals

- A. Deliège, A. Cioppa, and M. Van Droogenbroeck. Ghost loss to question the reliability of training data. *IEEE Access*, 8:44774–44782, March 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/245659.

- A. Cioppa, M. Braham, and M. Van Droogenbroeck. Asynchronous semantic background subtraction. *Journal of Imaging*, 6(6):1–20, June 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/248665.
  Patent granted by the US Patent Office.

Publications in international conferences

- A. Cioppa, A. Deliège, and M. Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 1846–1855, Salt Lake City, Utah, USA, June 2018.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/222427.
  Best paper award at the 2018 CVsports workshop.

- A. Deliège, A. Cioppa, and M. Van Droogenbroeck. An effective hit-or-miss layer favoring feature interpretation as learned prototypes deformations. In *AAAI Conference on Artificial Intelligence, Workshop on Network Interpretability for Deep Learning*, pages 1–8, Honolulu, Hawaii, USA, January-February 2019.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/229746.

- A. Cioppa, A. Deliège, M. Istasse, C. De Vleeschouwer, and M. Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2505–2514, Long Beach, California, USA, June 2019.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/234413.
  Best paper award at the 2019 CVsports workshop.

- A. Cioppa, A. Deliège, N. Ul Huda, R. Gade, M. Van Droogenbroeck, and T. Moeslund. Multimodal and multiview distillation for real-time player detection on a football field. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 3846–3855, Seattle, Washington, USA, June 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/246668.
  Best paper award at the 2020 CVsports workshop.

- A. Cioppa, A. Deliège, S. Giancola, B. Ghanem, M. Van Droogenbroeck, R. Gade, and T. Moeslund. A context-aware loss function for action spotting in soccer videos. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13123–13133, Seattle, Washington, USA, June 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/241893.

- A. Cioppa, M. Van Droogenbroeck, and M. Braham. Real-time semantic background subtraction. In *IEEE International Conference on Image Processing (ICIP)*, pages 3214–3218, Abu Dhabi, United Arab Emirates, October 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/247775.
  Patent protected by the US Patent Office.

Other publication

- A. Deliège, A. Cioppa, and M. Van Droogenbroeck. HitNet: a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. *CoRR*, abs/1806.06519, June 2018.

  PEER-REVIEWED: ✘, URL: https://orbi.uliege.be/handle/2268/225615.

Patent

- M. Van Droogenbroeck, M. Braham, and A. Cioppa. Foreground and background detection method. US Patent Office, July 2020.

  PEER-REVIEWED: ✔, URL: https://orbi.uliege.be/handle/2268/238274.
  STATUS: **B1** (accepted)

Part I
Low-level semantics

CHAPTER 2

Combining motion detection with semantic segmentation in real time

Contents

In Chapter 1, we have defined the different levels of semantics and presented some of their potential applications. This chapter presents a novel method, producing a first type of low-level semantics that describes which pixels belong to objects that are in motion in a video sequence. This task is usually called *background subtraction* or motion detection.

The current state-of-the-art unsupervised method for background subtraction, called Semantic Background Subtraction (SBS), relies on the combination of a background subtraction algorithm with a semantic segmentation algorithm. While SBS has been shown to improve background subtraction, a major difficulty is that it combines two streams generated at different frame rates. This results in SBS operating at the slowest frame rate of the two streams, usually being the one of the semantic segmentation algorithm.

Therefore, we propose a novel method, referred to as "Asynchronous Semantic Background Subtraction" (ASBS), able to combine a semantic segmentation algorithm with any background subtraction algorithm asynchronously. It achieves performances close to that of SBS while operating at the fastest possible frame rate, being the one of the

background subtraction algorithm. Our method consists in analyzing the temporal evolution of pixel features to possibly replicate the decisions previously enforced by semantics when no semantic segmentation is computed. We showcase ASBS with several background subtraction algorithms and also add a feedback mechanism that feeds the background model of the background subtraction algorithm to upgrade its updating strategy and, consequently, enhance the decision.

Experiments show that we systematically improve the performance, even when the semantic segmentation stream has a much slower frame rate than the frame rate of the background subtraction algorithm. Finally, we show that ASBS coupled with ViBe sets a new state of the art for real-time background subtraction algorithms and even competes with the non real-time state-of-the-art ones.

This chapter is organized as follows. Section 2.1 introduces the tasks of background subtraction and semantic segmentation as well as the challenges that we propose to solve in this chapter. These two tasks are used later on in other methods presented this thesis. Section 2.2 describes the semantic background subtraction (SBS) method that underpins our developments. In Section 2.3, we provide the details of our new method and introduce a semantic feedback mechanism to further improve the performances. Experimental results are provided in Section 2.4, and compared with those of the original semantic background subtraction method when semantics is missing for some frames. Finally, we conclude on this method in Section 2.5.

PUBLICATIONS RELATED TO THIS CHAPTER

A. Cioppa, M. Braham, and M. Van Droogenbroeck. Asynchronous semantic background subtraction. *Journal of Imaging*, 6(6):1–20, June 2020

A. Cioppa, M. Van Droogenbroeck, and M. Braham. Real-time semantic background subtraction. In *IEEE International Conference on Image Processing (ICIP)*, pages 3214–3218, Abu Dhabi, United Arab Emirates, October 2020

Contributions. **(i)** We propose a novel method for the task of background subtraction. **(ii)** We alleviate the problem of the slow computation of semantic segmentation by substituting it for some frames with the help of a newly introduced change detection algorithm. This makes our method usable in real time. **(iii)** We show that at a semantic frame rate corresponding to real-time computations, we achieve top performances, meaning that our substitute for semantics is adequate. **(iv)** We show real-time state-of-the-art performances on the CDNet 2014 dataset.

2.1 Introduction

The goal of **background subtraction** (shortened to BGS in the following) algorithms is to automatically segment moving objects in video sequences using a background model fed with features, hand-designed or learned by a machine learning algorithm, generally

computed for each video frame. Then, the features of the current frame are compared to the features of the background model to classify pixels either in the background or in the foreground. While being fast, these techniques remain sensitive to illumination changes, dynamic backgrounds, or shadows that are often segmented as moving objects.

Background subtraction has been an active field of research during the last years [18]. It was promoted by the development of numerous variations of the GMM [179] and KDE [54] algorithms, and the emergence of innovative algorithms such as SOBS [129], ViBe [13], SuBSENSE [176], PAWCS [177], IUTIS-5 [16], and PCA variants [52, 99]. Research in this field can count on large datasets annotated with ground-truth data such as the BMC dataset [191], the CDNet 2014 dataset [199], or the LASIESTA dataset [40], which was an incentive to develop supervised algorithms. In [23], Braham and Van Droogenbroeck were the first to propose a background subtraction method using a deep neural network; this work paved the way to other methods, proposed recently [19, 119, 200, 223]. Methods based on deep learning have better segmentation performances, but they rely on the availability of a fair amount of annotated training data; to some extent, they have lost their ability to deal with any camera operating in an unknown environment. Note however that, in their seminal work [23], Braham and Van Droogenbroeck present a variation of the network that is trained on ground-truth data generated by an unsupervised algorithm, thus requiring no annotations at all; this idea was later reused by Babaee *et al.* [8]. Rather than building novel complicated methods to overcome problems related to challenging operational conditions such as illumination changes, dynamic backgrounds, the presence of ghosts, shadows, camouflage or camera jitter, another possibility consists in leveraging low-level semantics provided by a universal semantic segmentation algorithm for improving existing BGS algorithms.

Semantic segmentation of images consists in labeling each pixel of an image with the class of its enclosing object or region. It is a well-covered area of research, but it is only recently that it has achieved the level of performance needed for real applications thanks to the availability of large annotated datasets such as ADE20K [225], Pascal VOC2012 [55], COCO [123] or Cityscapes [39], and novel deep neural networks [73, 78, 221]. The performances achieved by these deep networks for the task of semantic segmentation have motivated their use for various computer vision tasks such as optical flow computation [170], or motion segmentation [154, 194]. The underlying idea is to segment objects and characterize their motion using, in our case, background subtraction in video sequences [22]. It is important to note that semantic segmentation networks are trained with annotated datasets that contain various types of objects, most of which do not appear in videos such as those of the CDNet 2014 dataset. In other words, semantic segmentation algorithms are not tailored for the task of motion detection. While this is a suitable feature to deal with arbitrary unknown scenes, it requires to validate if a network works well on the typical images encountered in background subtraction.

Recently, Braham *et al.* [22] presented the semantic background subtraction method (named SBS hereafter), that leverages the output of semantic segmentation networks,

thanks to a custom *semantic classifier*, for improving background subtraction algorithms. This method, which combines the decision of two classifiers, namely the output of a background subtraction algorithm and a semantic classifier, reduces the mean error rate up to 20% for the 5 best unsupervised algorithms on CDNet 2014 [199], making it the state of the art across all unsupervised BGS algorithms. Unfortunately, in practice, it is often much slower to compute semantic segmentation than it is to perform background subtraction. Consequently, to avoid reducing the frame rate of the images processed by background subtraction, semantic segmentation needs to be computed on a dedicated hardware (such as a modern GPU) and fed asynchronously, that is with missing *semantic frames*.

To better understand the problem, let us analyze the timing diagram of SBS, as displayed in Figure 2.1. For this time analysis, we assume that a GPU is used for semantic

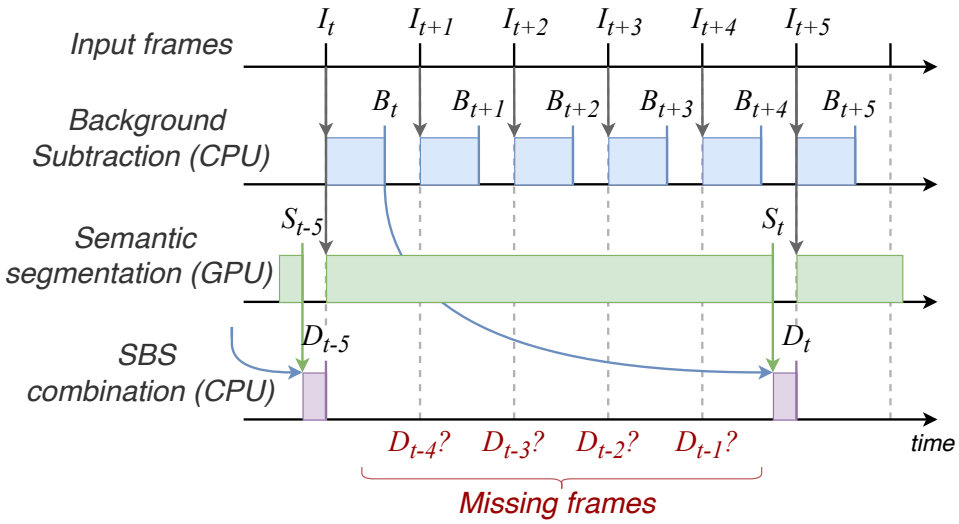


Figure 2.1: Timing diagram of a naive real-time implementation of the semantic background subtraction (SBS) method when the frame rate of the semantic segmentation network is too slow to handle all the frames in real time. From top to bottom, the time lines represent: the input frames I_t, the computation of intermediate motion masks B_t by the BGS algorithm (on CPU), the computation of semantic segmentation S_t by the semantic segmentation algorithm (on GPU), and the computation of output motion masks D_t by the SBS method (on CPU). Vertical lines indicate when an image is available and filled rectangular areas display when a GPU or CPU performs a task. Arrows show the inputs required by the different tasks. This diagram shows that even when the background subtraction algorithm is real time with respect to the input frame rate, it is the computation of semantic segmentation that dictates the output frame rate, leading to missing frames.

segmentation, and a CPU is used for both the BGS algorithm and the SBS method. When the GPU is available, it starts segmenting the input frame, otherwise it skips it. In the sce-

nario of a BGS algorithm being faster than the semantic segmentation algorithm, which is the scenario that we examine in this chapter, the BGS algorithm starts as soon as the previous processing is over. The CPU then waits until semantic segmentation has been computed and a semantic frame S_t is available. The timeline analysis of SBS shows that: (1) with respect to the input frame, the output frame is delayed by the time to compute semantic segmentation and to process the segmentation map (this delay is unavoidable and constant), and (2) the output frame rate is mainly driven by the slowest operation. It results that some output frames would be skipped, although the CPU computes all the intermediate masks by the BGS algorithm. For example, in the case of Figure 2.1, it is possible to apply the BGS algorithm to I_{t+2}, but not to process B_{t+2} with the help of the semantic classifier. In other words, the slowest operation dictates its rhythm (expressed in terms of frame rate) to the entire processing chain. Hence, the semantic segmentation network and the output have equal frame rates. This is not a problem as long as the output frame rate (or equivalently that of semantic segmentation) is faster than the input frame rate. However, the semantic segmentation frame rate is generally slower than the input frame rate, which means that it is not possible to process the video at its full frame rate, or in order words, that the processing of SBS is not real time.

To increase the output frame rate to its nominal value, we need to either accelerate the production of semantic segmentation maps, which induces the choice of a faster but less accurate semantic segmentation network, or to interpolate the missing semantic frames. Our analysis on semantic segmentation networks showed that faster networks are not exploitable because of their lack of accuracy. Also, semantic segmentation networks should be preferred to instance segmentation networks. For example, we had to discard MaskRCNN [78] and prefer the PSPNet network [221], as shown in Table 2.1. An

Table 2.1: Comparison of the best mean F_1 score achieved for two semantic segmentation networks used in combination with SBS on the CDNet 2014 dataset. These performances are obtained considering the SBS method, where the output of the BGS algorithm is replaced by the ground-truth masks. This indicates how the semantic segmentation used in SBS would deteriorate a perfect BGS algorithm.

Networks	SBS with PSPNet [221]	SBS with MaskRCNN [78]
Best mean F_1	0.953	0.674

alternative option is to interpolate missing semantic frames. Naive ideas would be to skip the SBS processing step in the absence of semantic segmentation or to systematically repeat the last pixelwise semantic segmentation information when it is missing. Both ideas proved unsuccessful, as shown in our experiments (see Section 2.4.2). A better idea is to avoid any mechanism that would substitute itself to the difficult calculation of semantic segmentation and, instead, replicate the decisions enforced previously by the semantic classifier to compensate for the lack of semantic segmentation later on when relevant. The underlying question is whether or not we should trust and repeat decisions taken by SBS [22].

2.2 Description of the semantic background subtraction method

SBS combines the decision of two classifiers that operate at each pixel (x, y) and for each frame (indexed by t): (1) a background subtraction algorithm, which is a binary classifier between the background (BG) and the foreground (FG), whose output is denoted by $B_t(x, y) \in \{\text{BG}, \text{FG}\}$, and (2) a semantic three-class classifier, whose output is denoted by $S_t(x, y) \in \{\text{BG}, \text{FG}, \text{"?"}\}$, where the third class, called the "don't know" class and denoted by "?", corresponds to cases where the semantic classifier is not able to take a decision. This semantic classifier is built upon two signals that contribute to take a decision. The first one is the semantic probability that pixel (x, y) belongs to a set of objects most likely in motion. If this signal is lower than some threshold τ_{BG}, $S_t(x, y)$ is set to BG. The second signal is a pixelwise increment of semantic probability for a pixel (x, y) to belong to a moving object. When this signal is larger than another threshold τ_{FG}, $S_t(x, y)$ is set to FG. In all other cases, $S_t(x, y)$ is undetermined and is assigned a "don't know" "?" class.

Finally, the output of SBS, noted $D_t(x, y)$, is a combination of $B_t(x, y)$ and $S_t(x, y)$, as outlined in Table 2.2. This combination works as follows: when $S_t(x, y)$ is determined (either BG or FG), this class is chosen as the output of SBS regardless of the value of $B_t(x, y)$; when $S_t(x, y)$ is undetermined (which corresponds to "?" cases), the class of $B_t(x, y)$ is chosen as $D_t(x, y)$.

	Decision table of SBS		
	Classifiers		**Output**
	$B_t(x, y)$	$S_t(x, y)$	$D_t(x, y)$
(L1)	BG	"?"	BG
(L2)	BG	BG	BG
(L3)	BG	FG	FG
(L4)	FG	"?"	FG
(L5)	FG	BG	BG
(L6)	FG	FG	FG

Table 2.2: Decision table for the output of SBS ($D_t(x, y)$) based on the output of two classifiers: a BGS algorithm ($B_t(x, y)$) and a semantic classifier ($S_t(x, y)$).

While SBS is effective to handle challenging BGS scenarios, it can only be real time if both classifiers are real time. As the decision of the semantic classifier supersedes that of $B_t(x, y)$ in two scenarios (see lines (L3) and (L5) in Table 2.2), it is essential to rely on a high-quality semantic segmentation, which is not achievable with faster semantic networks.

Another way to reduce the computation time of semantic information is to segment small portions of the image or to skip some frames. However, according to the original decision table of SBS, this would introduce more "don't know" cases for the semantic

classifier (equivalent to lines (L1) and (L4) of Table 2.2). Our method presented in the next section aims at providing a decision different from the "don't know" case when the semantic segmentation has not been calculated for pixel (x, y) at time t.

2.3 Asynchronous semantic background subtraction

2.3.1 Description of the method

We propose a novel method that reuses previous decisions of the semantic classifier in the absence of semantic segmentation. We choose to rely on this previous decision and check whether or not this decision is still relevant. If the pixel has not changed too much, its predicted semantic class is still likely untouched, and therefore the previous decision of the semantic classifier is replicated.

Technically, we introduce a third classifier in the previous decision table. This classifier corresponds to a change detection algorithm whose task is to predict whether or not a pixel's value has significantly changed between the current image, at time t, and the last time semantic information was available for that pixel, at time $t^* \leq t$. The new decision table is presented in Table 2.3 and works as follows: if the change detection algorithm, whose output is denoted by $C_t(x, y)$, predicts that the pixel has not changed, it means that the pixel still probably belongs to the same object and thus the previous decision of the semantic classifier is repeated. Alternatively, when the change detection algorithm predicts that the pixel has changed too much, the previous decision of the semantic classifier cannot be trusted anymore, leaving it to the BGS algorithm to classify the pixel. The improvement of our algorithm compared to SBS originates from lines (L3) and (L6) of Table 2.3, as without the change detection classifier, the final decision would be taken by the BGS algorithm alone.

The only requirement for the choice of the change detection algorithm is that it has to be real time. Therefore, we choose a simple yet effective algorithm that relies on the Manhattan distance between the current pixel's color value and its previous color value when semantic segmentation was last available, at time t^*. If this color distance is smaller (respectively larger) than some threshold, the change detection algorithm predicts that the pixel has not changed (respectively has changed). Let us note that we use two different thresholds depending on the output of the semantic classifier (τ^*_{BG} if $S_{t^*}(x, y) = \text{BG}$, or τ^*_{FG} if $S_{t^*}(x, y) = \text{FG}$) since the foreground objects and the background change at different rates. In the case where semantic segmentation is available, the change detection algorithm will obviously always predict that the pixel has not changed since $t^* = t$, and the decision table of ASBS (Table 2.3) degenerates into that of SBS (Table 2.2).

	Decision table of ASBS			
	Classifiers			**Output**
	$B_t(x,y)$	$S_{t^*}(x,y)$	$C_t(x,y)$	$D_t(x,y)$
(L1)	BG	"?"	"✖"	BG
(L2)	BG	BG	"✖"	BG
(L3)	BG	FG	No Change	FG
(L4)	BG	FG	Change	BG
(L5)	FG	"?"	"✖"	FG
(L6)	FG	BG	No Change	BG
(L7)	FG	BG	Change	FG
(L8)	FG	FG	"✖"	FG

Table 2.3: Decision table of ASBS. Its output ($D_t(x,y)$) depends on three classifiers: a BGS algorithm ($B_t(x,y)$), information about the last time, $t^* \leq t$, the semantic classifier ($S_{t^*}(x,y)$) classified the pixel, and a change detection algorithm ($C_t(x,y)$). The "don't care" values ("✖") represent cases where $C_t(x,y)$ has not impact on $D_t(x,y)$, either because previous semantic information is undetermined or because $B_t(x,y)$ and $S_{t^*}(x,y)$ agree on the class.

2.3.2 Timing diagram

The ASBS method introduces a small computational overhead with the change detection algorithm. However, this overhead is negligible with respect to the computation of semantic segmentation. The practical benefits of ASBS can be visualized on a detailed timing diagram of its components. For a formal discussion, we use the following notations:

- I_t, B_t, S_t, C_t D_t respectively denote an arbitrary input frame, the background segmented by the BGS algorithm, semantic segmentation, the output of the change detection algorithm, and the background segmented by ASBS, indexed by t.
- δ_I represents the time between two consecutive input frames.
- Δ_B, Δ_S, Δ_C Δ_D are the times needed to calculate the BGS output, the semantic segmentation, the change detection output, and to apply SBS or ASBS, which are supposed to be the same, respectively. These times are reasonably constant.

We assume that semantic segmentation is calculated on a GPU, whereas the BGS and the application of the rules are calculated on a single threaded CPU hardware. Also, the frame rate of semantic segmentation is supposed to be smaller than that of BGS; that is $\Delta_S > \Delta_B$.

We now examine two different scenarios. The first scenario is that of a real-time BGS algorithm ($\Delta_B < \delta_I$) satisfying the condition $\Delta_B + \Delta_D < \delta_I$. This scenario, illustrated in Figure 2.2, can be obtained with the ViBe [13] BGS algorithm for example. On the timing

diagram, it can be seen that the output frame rate is then equal to the input frame rate, all frames being segmented either by the SBS decision table or ASBS decision table with a time delay corresponding approximately to $\Delta_S + \Delta_D$. We present illustrative numbers for this timing diagram in Section 2.4.4.

In a second scenario, the frame rate of the BGS is too slow to accommodate to real time with ASBS. It means that $\Delta_B + \Delta_D > \delta_I$. In this case, the output frame rate is mainly dictated by Δ_B, since $\Delta_B \gg \Delta_D$. The input frame rate can then be viewed as slowed down by the BGS algorithm, in which case the timing diagrams fall back to the same case as a real-time BGS algorithm by artificially changing δ_I to $\tilde{\delta}_I$, where $\tilde{\delta}_I = \Delta_B + \Delta_D > \delta_I$. It is a scenario that, unfortunately, follows the current trend to produce better BGS algorithms at the price of more complexity and lower processing frame rates. Indeed, according to our experiments and [162], the top unsupervised BGS algorithms ranked on the CDNet web site (see http://changedetection.net) are not real time. Note that Δ_C has no impact on the previous timings, as long as $\Delta_C < \Delta_B$, which is the case in all of our setups.

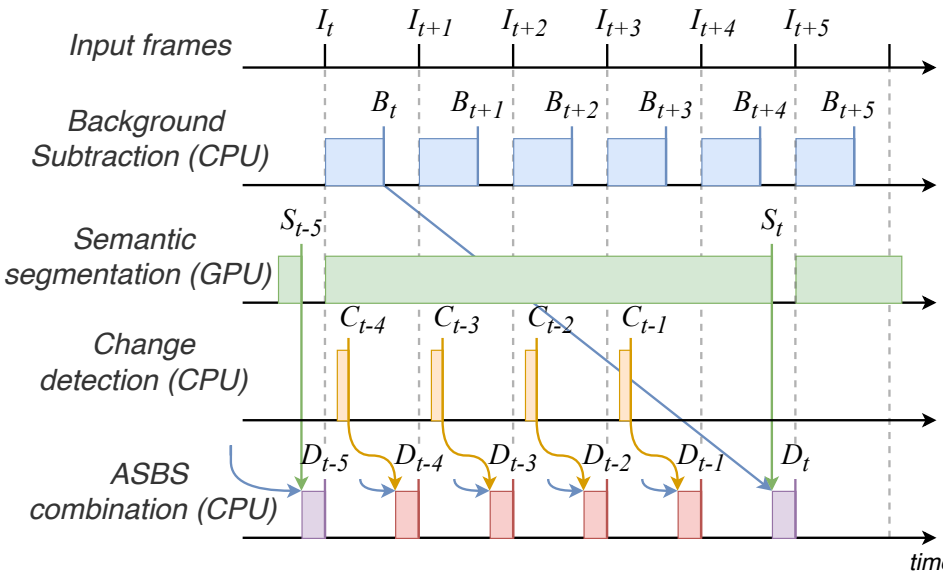


Figure 2.2: Timing diagram of ASBS in the case of a real-time BGS algorithm ($\Delta_B < \delta_I$) satisfying the condition $\Delta_B + \Delta_D < \delta_I$. Note that the output stream is delayed by a constant $\Delta_S + \Delta_D$ time with respect to the input stream.

2.3.3 Introducing a semantic feedback

The methods SBS and ASBS are designed to be combined to a BGS algorithm to improve the quality of its final motion segmentation, but they do not affect the decisions taken by the BGS algorithm itself. In this section, we explore possibilities to embed information provided by the semantic classifier inside the BGS algorithm itself, which would remain blind to it otherwise. Obviously, this requires to craft modifications specific to a particular algorithm or family of algorithms, which can be effortful as explained hereinafter.

The backbone of many BGS algorithms is composed of three main parts. First, an internal model of the background is kept in memory, for instance in the form of color samples or other types of features. Second, the input frame is compared to this model via a distance function to classify pixels as background or foreground. Third, the background model is updated to account for changes in the background over time.

A first possibility to embed semantic information inside the BGS algorithm is to include the semantic segmentation map directly in a joint background model integrating color and semantic segmentation features. This requires to formulate the relationships that could exist between them and to design a distance function accounting for these relationships, which is not trivial. Therefore, we propose a second way of doing so by incorporating this semantic information during the update, which is straightforward for algorithms whose model updating policy is conservative (as introduced in [13]). For those algorithms, the background model in pixel (x, y) may be updated if $B_t(x, y) = \text{BG}$, but it is always left unchanged if $B_t(x, y) = \text{FG}$, which prevents the background model from being corrupted with foreground features. In other words, the motion segmentation map B_t serves as an updating mask. As D_t produced by SBS or ASBS is an improved version of B_t, we can advantageously use D_t instead of B_t to update the background model, as illustrated in Figure 2.3. This introduces a semantic feedback (via lines (L3) and (L6) of Table 2.3) which improves the internal background model and, consequently, the next segmentation map B_{t+1}, whether or not semantic segmentation is computed.

2.4 Experiments

In this section, we evaluate the performances of our novel method ASBS and compare them to those of the original BGS algorithm and those of the original SBS method [22]. First, in Section 2.4.1, we present our evaluation methodology. This comprises the choice of a dataset along with the evaluation metric, and all needed implementation details about ASBS, such as how we compute the semantic segmentation, and how we choose the values of the different thresholds. In Section 2.4.2, we evaluate ASBS when combined with state-of-the-art BGS algorithms. Section 2.4.3 is devoted to a possible variant of ASBS which includes a feedback mechanism that can be applied to any conservative BGS algorithm. Finally, we discuss the computation time of ASBS in Section 2.4.4.

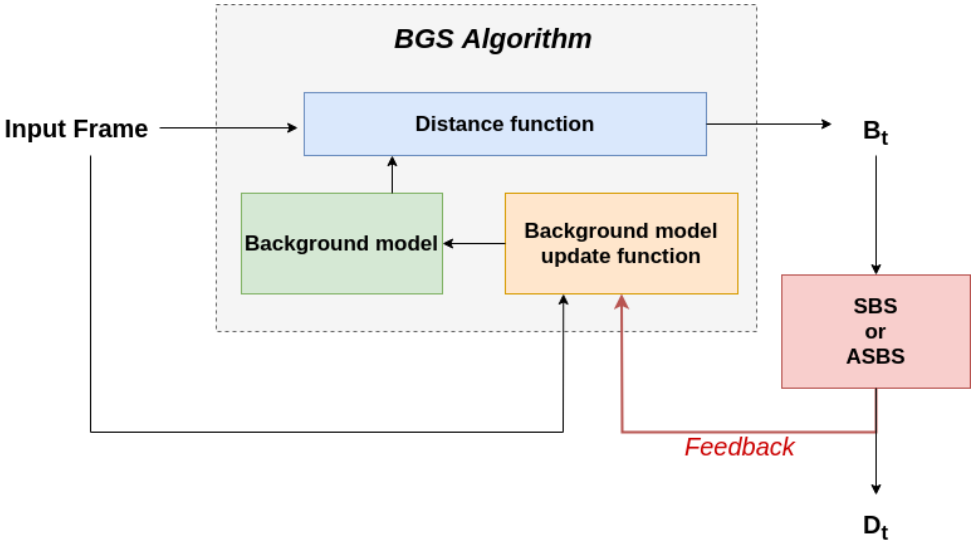


Figure 2.3: Our feedback mechanism, which impacts the decisions of any BGS algorithm whose model update is conservative, consists to replace the BG/FG segmentation of the BGS algorithm by the final segmentation map improved by semantics (either by SBS or ASBS) to update the internal background model.

2.4.1 Evaluation methodology

For the quantitative evaluation, we chose the CDNet 2014 dataset [199] which is composed of 53 video sequences taken in various environmental conditions such as bad weather, dynamic backgrounds and night conditions, as well as different video acquisition conditions, such as PTZ and low frame rate cameras. This challenging dataset is largely employed within the background subtraction community and currently serves as the reference dataset to compare state-the-art BGS techniques.

We compare performances on this dataset according to the overall F_1 score, which is one of the most widely used performance score for this dataset. For each video, F_1 is computed by:

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \tag{2.1}$$

where TP (true positives) is the number of foreground pixels correctly classified, FP (false positives) the number of background pixels incorrectly classified, and FN (false negatives) the number of foreground pixels incorrectly classified. The overall F_1 score on the entire dataset is obtained by first averaging the F1 scores over the videos, then over the categories, according the common practice of CDNet [199]. Note that this averaging in-

troduces inconsistencies between overall scores that can be avoided by using summarization instead, as described in [144], but to allow a fair comparison with the other BGS algorithms, we decide to stick to the original practice of [199] for our experiments.

We compute the semantic segmentation as in [22], that is with the semantic segmentation network PSPNet [221] trained on the ADE20K dataset [224] (using the public implementation [220]). The network outputs a vector containing 150 real numbers for each pixel, where each number is associated to a particular object class within a set of 150 mutually exclusive classes. The semantic probability estimate $p_{S,t}(x,y)$ is computed by applying a softmax function to this vector and summing the values obtained for classes that belong to a subset of classes that are relevant for motion detection. We use the same subset of classes as in [22] (person, car, cushion, box, boot, boat, bus, truck, bottle, van, bag and bicycle), whose elements correspond to moving objects of the CDNet 2014 dataset.

For dealing with missing semantic segmentation, we have restricted the study to the case of a temporal sub-sampling of one semantic frame per X original frames; this sub-sampling factor is referred to as X:1 hereafter. In other scenarios, semantic segmentation could be obtained at a variable frame rate or for some variable regions of interest, or even a mix of these sub-sampling schemes.

The four thresholds are chosen as follows. For each BGS algorithm, we optimize the thresholds $(\tau_{\mathrm{BG}}, \tau_{\mathrm{FG}})$ of SBS with a grid search to maximize its overall F_1 score. Then, in a second time, we freeze the optimal thresholds $(\tau^{\circ}_{BG}, \tau^{\circ}_{FG})$ found by the first grid search and optimize the thresholds $(\tau^{*}_{BG}, \tau^{*}_{FG})$ of ASBS by a second grid search for each pair (BGS algorithm, X:1), to maximize the overall F_1 score once again. Such methodology allows a fair comparison between SBS and ASBS as the two techniques use the same common parameters $(\tau^{\circ}_{BG}, \tau^{\circ}_{FG})$ and ASBS is compared to an optimal SBS method. Note that the α parameter is chosen as in [22] since it does not significantly affect the performances.

The segmentation maps of the BGS algorithms are either taken directly from the CDNet 2014 website (when no feedback mechanism is applied) or computed using the public implementations available at [11] for ViBe [13] (when the feedback mechanism of Section 2.4.3 is applied).

2.4.2 Performances of our method

A comparison of the performances obtained with SBS and ASBS for four state-of-the-art BGS algorithms (IUTIS-5 [16], PAWCS [177], SuBSENSE [176], and WebSamBe [101]) and for different sub-sampling factors is provided in Figure 2.4. For the comparison with SBS, we used two heuristics that can also extend SBS in the case of missing semantic segmentation. The first heuristic never repeats the decision of the semantic classifier and the second heuristic always repeats the decision of the semantic classifier without checking if the pixel's value has changed. Note that the former corresponds to ASBS with $\tau^{*}_{BG} < 0$ and $\tau^{*}_{FG} < 0$ and the latter with τ^{*}_{BG} and τ^{*}_{FG} chosen larger than the upper bound

of color distances.

As can be seen, the performances of ASBS decrease much more slowly than those of SBS with the decrease of the semantic frame rate and, therefore, are much closer to those of the ideal case (SBS with all semantic maps computed, that is ASBS 1:1), meaning that ASBS provides better decisions for frames without semantics.

A second observation can be made concerning the second heuristic. The performances become worse than the ones of the original BGS for semantic frame rates lower than 1 out of 5 frames, but they are better than the first heuristic for high semantic frame rates. This observation emphasizes the importance of checking the change detection algorithm of ASBS instead of blindly repeating the corrections induced by the semantic classifier. Its performances for lower frame rates are not represented for the sake of figure clarity but still decrease linearly to very low performances. For example, in the case of IUTIS_5, the performance drops to 0.67 at 25:1. Finally, it can be seen that, on average, ASBS with 1 frame of semantics out of 25 frames (ASBS 25:1) performs as well as the first heuristic, with 1 frame of semantics out of 2 frames.

In Figure 2.5, we also compare the effects of the first heuristic for frames with missing semantic segmentation, and ASBS for different BGS algorithms by looking at their performances in the mean ROC space of CDNet 2014 (ROC space where the false and true foreground rates are computed according to the rules of [199]). The points represent the performances of different BGS algorithms whose segmentation maps can be downloaded on the dataset website. The arrows represent the effects of SBS and ASBS for a temporal sub-sampling factor of 5:1. This choice of frame rate is motivated by the fact that it is the frame rate at which PSPNet can produce the segmentation maps on a GeForce GTX Titan X GPU. We observe that SBS improves the performances, but only marginally, whereas ASBS moves the performances much closer to the oracle (upper left corner).

To better appreciate the positive impact of our strategy for replacing the decision of the semantic classifier, we also provide a comparative analysis of the F_1 score by only considering the frames without semantic segmentation. We evaluate the relative improvement of the F_1 score of ASBS, SBS and the second heuristic compared to the original BGS algorithm (which is equivalent to the first heuristic in this case). In Figure 2.6, we present our analysis on a per-category basis, in the same fashion as in [22]. As shown, the performances of ASBS are close to the ones of SBS for almost all categories, indicating that our substitute for semantic segmentation is adequate. We can also observe that the second heuristic does not perform well, and often degrades the results compared the original BGS algorithm. In this Figure, SBS appears to fail for two categories: “night videos” and “thermal”. This results from the ineffectiveness of PSPNet to process videos of these categories, as this network is not trained with such image types. Interestingly, ASBS is less impacted than SBS because it refrains from copying some wrong decisions enforced by the semantic classifier.

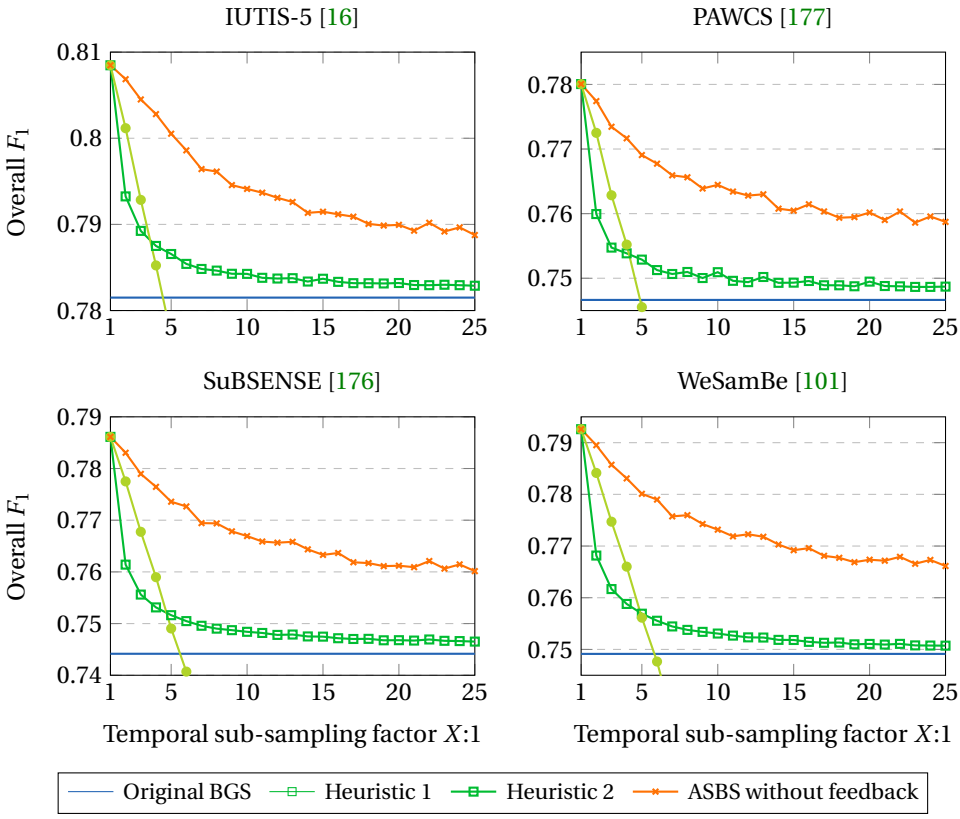
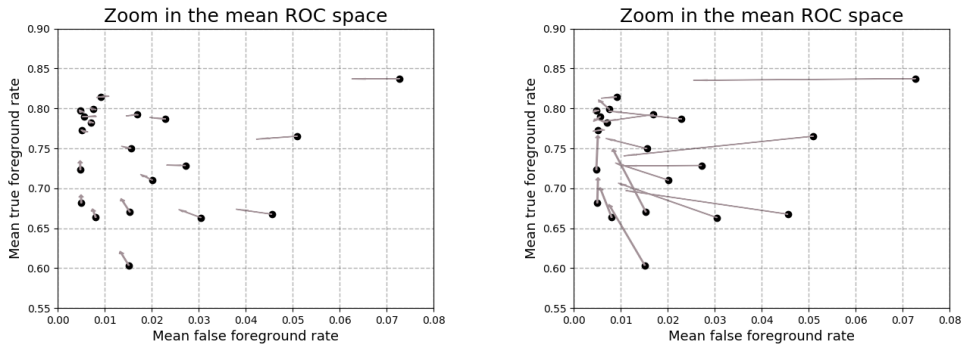


Figure 2.4: Overall F_1 scores obtained with SBS and ASBS for four state-of-the-art BGS algorithms and different sub-sampling factors. The performances of ASBS decrease much more slowly than those of SBS with the decrease of the semantic frame rate and, therefore, are much closer to those of the ideal case (SBS with all semantic segmentation maps computed, that is ASBS 1:1), meaning that ASBS provides better decisions for frames in those missing cases. On average, ASBS with 1 frame of semantic segmentation out of 25 frames (ASBS 25:1) performs as well as the first heuristic with 1 frame of semantic segmentation out of 2 frames (Heuristic 1 2:1).



(a) SBS at 5:1 (b) ASBS at 5:1 (ours)

Figure 2.5: Effects of SBS and ASBS on BGS algorithms in the mean ROC space of CDNet 2014 [199]. Each point represents the performance of a BGS algorithm and the end of the associated arrow indicates the performance after application of the methods for a temporal subsampling factor of 5:1. We observe that SBS (through the first heuristic) improves the performances, but only marginally, whereas ASBS moves the performances much closer to the oracle (upper left corner).

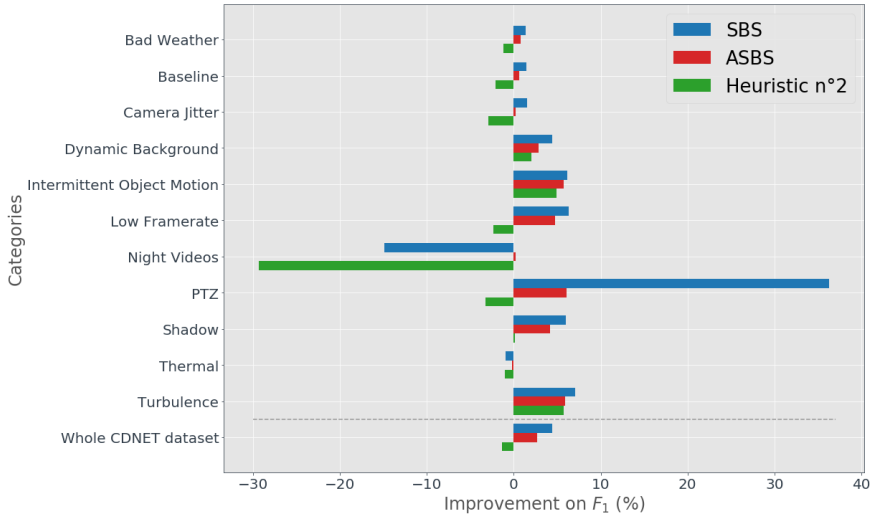


Figure 2.6: Per-category analysis. We display the relative improvements of the F_1 score of SBS, ASBS, and the second heuristic compared with the original algorithms, by considering only the frames without semantic segmentation (at a 5:1 semantic frame rate).

Finally, in Figure 2.7, we provide the evolution of the optimal parameters τ^*_{BG} and τ^*_{FG} with the temporal sub-sampling factor (in the case of PAWCS). The optimal value decreases with the sub-sampling factor, implying that the matching condition on colors become tighter or, in other words, that the decision of the semantic classifier should be less trusted for lower semantic frame rates, as a consequence of the presence of more outdated color features in further images.

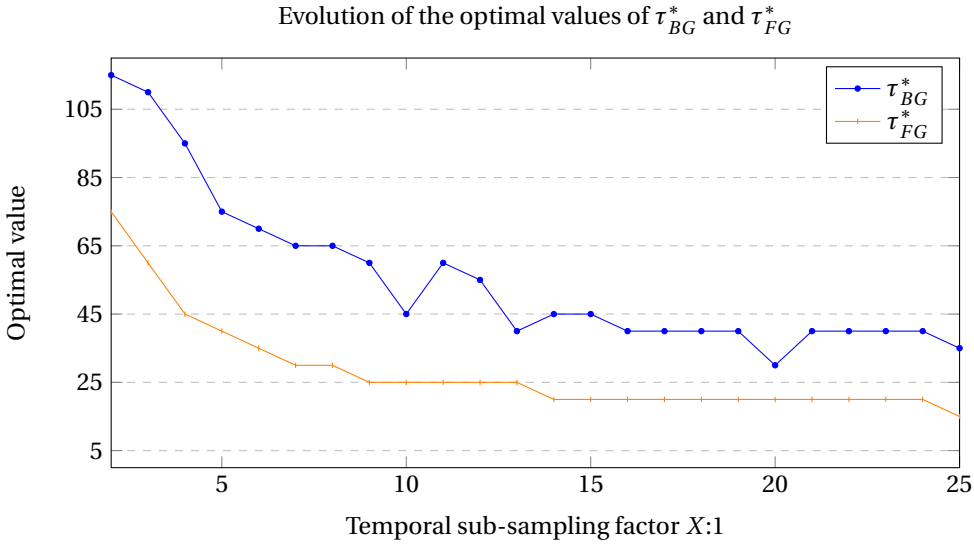


Figure 2.7: Evolution of the optimal thresholds τ^*_{BG} and τ^*_{FG} of the ASBS method when the semantic frame rate is reduced. Note that the Manhattan distance associated to these thresholds is computed on 8-bit color values. The results are shown here for the PAWCS algorithm, and follow the same trend for the other BGS algorithms considered in Figure 2.4.

2.4.3 A feedback mechanism

To appreciate the benefit of a semantic feedback, we performed experiments using the best real-time conservative BGS algorithm, ViBe [12]. Figure 2.8 (left column) reports the results of ASBS with the feedback mechanism on ViBe, and compares them to the original algorithm and the SBS method (through the first heuristic). Two main observations can be made. First, as for the results of the previous section, SBS and ASBS both improve the performances even when the semantic frame rate is low. Also, ASBS always performs better. Second, including the feedback always improves the performances for both SBS and ASBS. In the case of ViBe, the performance is much better when the feedback is included.

We also investigated to what extend the feedback provides better updating maps to the BGS algorithm. For conservative algorithms, this means that, internally, the background model is built with better features. This measure can be evaluated using the output of the classification map, B_t.

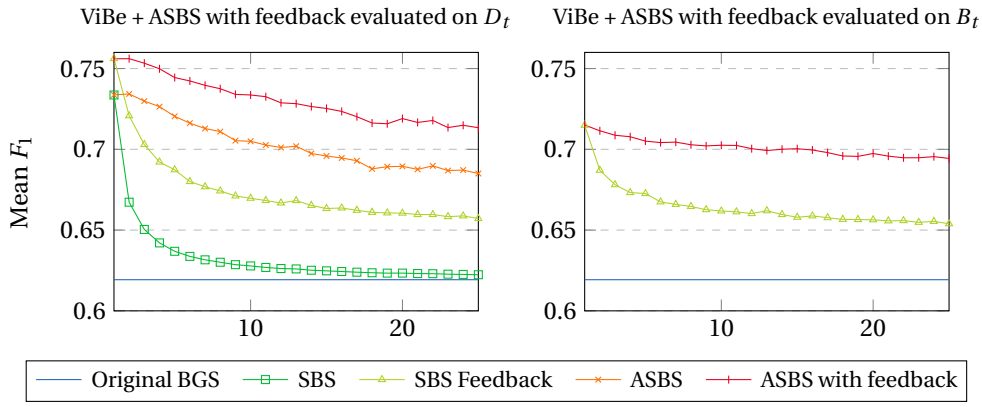


Figure 2.8: Comparison of the performances, computed with the overall F_1 score on the CDNet 2014, of SBS and ASBS when there is a feedback that uses D_t to update the model of the BGS algorithm. The results are given with respect to a decreasing semantic frame rate. It can be seen that SBS and ASBS always improve the results of the original BGS algorithm and that a feedback is beneficial. Graphs in the right column show that the intrinsic quality of the BGS algorithm is improved, as its output B_t, prior to any combination with semantics, produces higher mean F_1 scores.

For that purpose, we compared the original BGS algorithm and the direct output, that is B_t in Figure 2.3, of the feedback method when the updating map is replaced by D_t obtained by either SBS or ASBS. As can be seen in Figure 2.8 (right column), using the semantic feedback always improves the BGS algorithm whether the updating map is obtained from SBS or ASBS. This means that the internal background model of the BGS algorithm is always enhanced and that, consequently, a feedback helps the BGS algorithm to take better decisions in the first place.

Let us note that ViBe, which is a real-time BGS algorithm, combined with semantic segmentation provided at a real-time frame rate (about 1 out of 5 frames) and with the feedback from ASBS has an overall F_1 score of 0.746, which is the same performance as the original SuBSENSE algorithm (0.746) that is not real time [162]. It can be seen that our method can thus help real-time algorithms to reach performances of the top unsupervised BGS algorithms while meeting the real-time constraint, which is a huge advantage in practice.

In this same spirit, we compare this score with the top-5 state-of-the-art unsupervised background subtraction algorithms in Table 2.4. As can be seen, the performance

of our algorithm are comparable with the state-of-the-art algorithms which are not real time. Furthermore, our method performs better than all real-time unsupervised BGS algorithms, making it the state of the art for real-time unsupervised algorithms. We also performed a scene-specific Bayesian optimization [1] of the parameters; this leads to one set of parameters for each video. It corresponds to a more practical use of a BGS algorithm where its parameters are tuned for each application. With this particular optimization, we obtain an overall F_1 score of 0.828. This high score should not be compared with the others, but still shows the great potential of our algorithm in real-world applications. We showcase some motion detection masks of our novel methods in Figure 2.9 on one video of each category of the CDNet2014 dataset using ViBe as BGS algorithm.

Unsupervised BGS algorithms	F_1	fps
SemanticBGS (SBS with IUTIS-5) [22]	0.789	≈ 7
IUTIS-5 [16]	0.772	≈ 10
IUTIS-3 [16]	0.755	≈ 10
WisenetMD [76]	0.754	≈ 12
WeSamBE [101]	0.745	≈ 2
PAWCS [175]	0.740	$\approx 1-2$
ViBe [13]	0.619	≈ 152
ASBS at $X:5$ with ViBe and feedback	0.746	25
ASBS at $X:10$ with ViBe and feedback	0.734	50
ASBS at $X:5$ with ViBe and feedback and scene-specific optimization	**0.828**	**25**

Table 2.4: Comparison of the performance and speed of ASBS (build upon ViBe + feedback) with the top-5 unsupervised BGS algorithms on the CDNet 2014 dataset and the previous best real-time one. Our algorithm improves on some state-of-the-art algorithms while being real time and surpasses all real-time BGS algorithms. The mean frame rates (fps) are taken from [104].

2.4.4 Time analysis of our method

In this section, we show the timing diagram of ASBS and provide typical values for the different computation durations.

The timing diagram of ASBS with feedback is presented in Figure 2.10. The inclusion of a feedback has two effects. First, we need to include the feedback time Δ_F in the time needed for the background subtraction algorithm Δ_B. In our case, as we only substitute the updating map by D_t, it can be implemented as a simple pointer replacement and therefore Δ_F is negligible (in the following, we take $\Delta_F \simeq 0\,\text{ms}$). Second, we have to wait for the ASBS (or SBS) to finish before starting the background subtraction of the next frame.

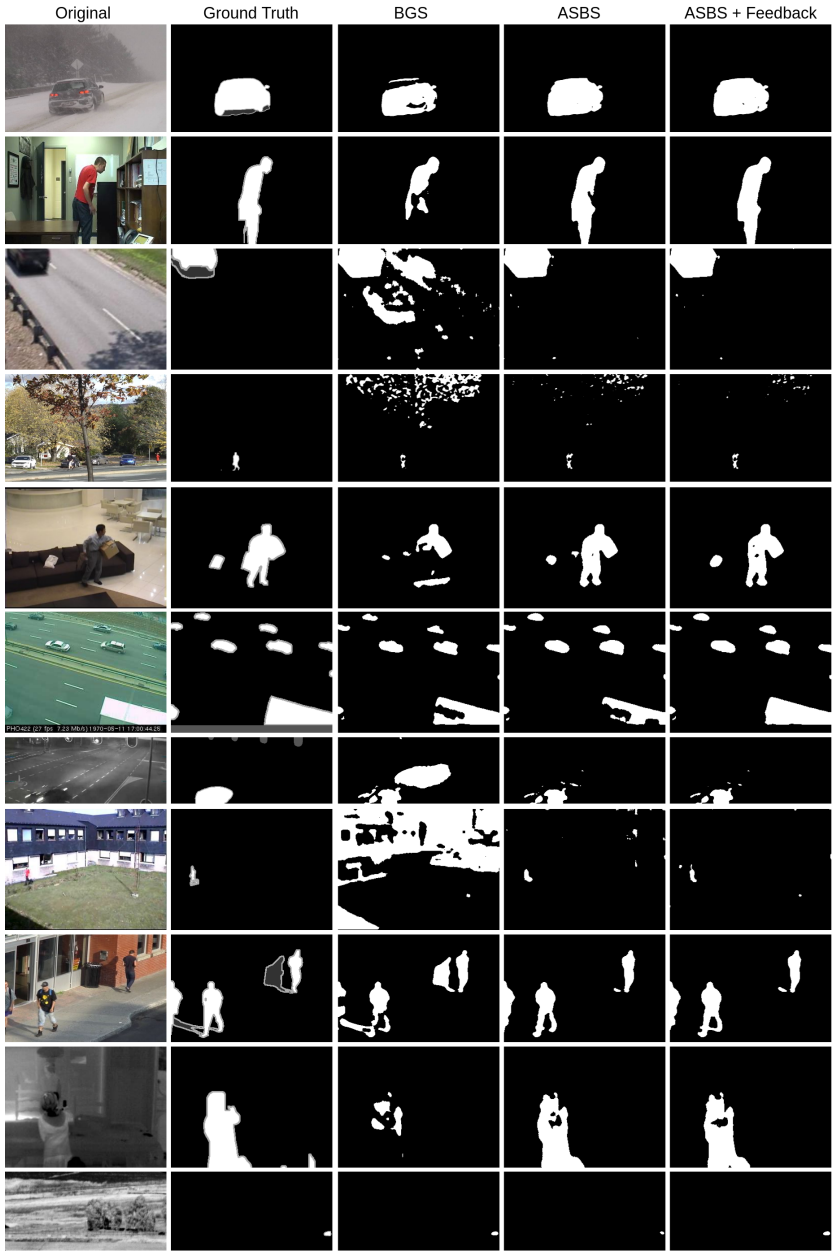


Figure 2.9: Illustration of the results of ASBS using ViBe as BGS algorithm. From left to right, we provide the original color image, the ground truth, the BGS as provided by the original ViBe algorithm, using our ASBS method without any feedback, and using ASBS and a feedback. Each line corresponds to a representative frame of a video in each category of CDNet2014.

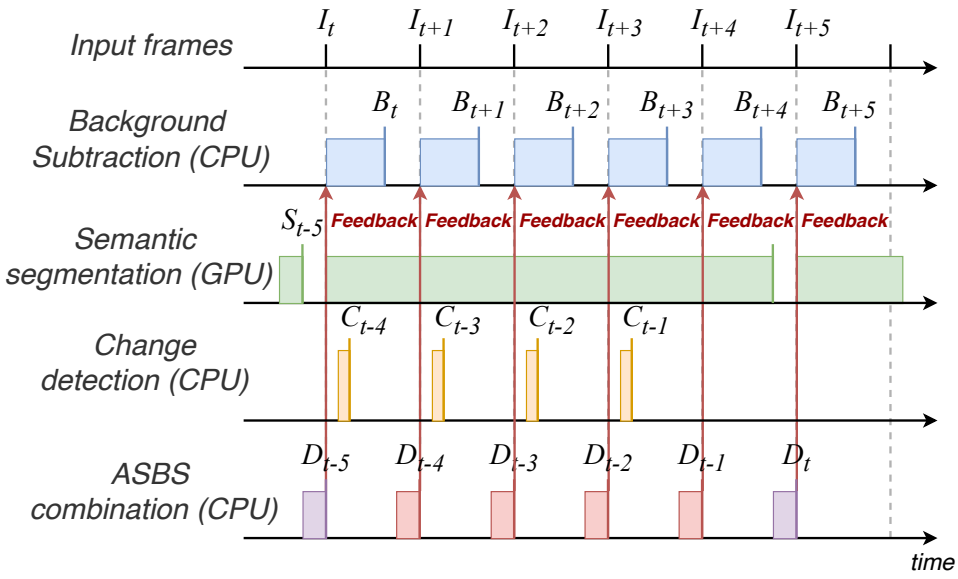


Figure 2.10: Timing diagram of ASBS with a feedback mechanism in the case of a real-time BGS algorithm ($\Delta_B < \delta_I$) satisfying the condition $\Delta_B + \Delta_D < \delta_I$ and the computation of semantic segmentation being not real-time ($\Delta_S > \delta_I$). Note that the feedback time Δ_F is negligible.

Concerning the computation time of BGS algorithms, Roy *et al.* [162] have provided a reliable estimate of the processing speed of leading unsupervised background subtraction algorithms. They show that the best performing ones are not real time. Only a handful of algorithms are actually real time, such as ViBe that can operate at about 200 fps on CDNet 2014 dataset, that is $\Delta_B = 5$ ms. With PSPNet, the semantic frame rate is of about 5 to 7 fps for a NVIDIA GeForce GTX Titan X GPU, which corresponds to $\Delta_S \simeq 200$ ms. It means that for 25 fps videos, we have access to semantics about once every 4 to 5 frames. In addition, Table 2.5 reports our observation about the mean execution time per frame of Δ_D for SBS and ASBS. These last tests were performed on a single thread running on a single processor Intel(R) Xeon(X) E5-2698 v4 2.20GHz.

Table 2.5: Mean computation time Δ_D (ms/frame) of SBS and ASBS.

Δ_D(SBS)	1.56
Δ_D(ASBS : frames with semantics)	2.12
Δ_D(ASBS : frames without semantics)	0.8

Thus, in the case of ViBe, we start from a frame rate of about 200 fps in its original version to reach about 160 fps when using ASBS, which is still real time. This is important

because, as shown in Section 2.4.3, the performances of ViBe with ASBS at a semantic frame rate of 1 out of 5 frames and feedback is the same as SuBSENSE that, alone, runs at a frame rate lower than 25 fps [162]. Hence, thanks to ASBS, we can replace BGS algorithms that work well but are too complex to run in real time and are often difficult to interpret by a combination of a much simpler BGS algorithm and a processing based on semantic segmentation, regardless of the frame rate of the last. Furthermore, ASBS is much easier to optimize as the parameters that we introduce are few in number and easy to interpret. In addition, we could also fine-tune the classes of the semantic segmentation algorithm, by selecting a dedicated set of objects to be considered, for a scene-specific setup. It is our belief that there are still some margins for further improvements.

2.5 Conclusion

In this chapter, we presented a novel method, named ASBS, based on semantic segmentation for improving the quality of motion segmentation masks produced by background subtraction algorithms when semantic segmentation is not computed for all video frames. ASBS, which is derived from the semantic background subtraction method, is applicable to any off-the-shelf background subtraction algorithm and introduces a novel decision table in order to repeat semantic decisions, even when semantics and the background are computed asynchronously. We also presented a feedback mechanism to update the background model with better samples and thus take better decisions. We showed that ASBS improves the quality of the motion segmentation masks compared to the original semantic background subtraction method applied only to frames with semantic segmentation masks. Furthermore, ASBS is straightforward to implement and cheap in terms of computation time and memory consumption. We also showed that applying ASBS with the feedback mechanism allows to elevate an unsupervised real-time background subtraction algorithm to the performance of non real-time state-of-the-art algorithms.

As a generalization, when semantic segmentation is missing for some frames but needed to help another task (in our case, the task of background subtraction), our method provides a convenient and effective mechanism to interpolate the missing semantic information. The mechanism of ASBS might thus enable real-time computer vision tasks requiring semantic information.

CHAPTER 3

Online distillation: basic principles applied to semantic segmentation

Contents

As seen in Chapter 2, high-quality semantic segmentation can be essential for improving background subtraction algorithms. It is also a very useful tool for global scene understanding in many areas, including sports as it can provide an accurate description of the objects in a scene and the environment in which they are. However, it has inherent difficulties, such as the absence of well-performing real-time networks or the need for pixel-wise annotated training data.

In this chapter, rather than substituting semantic segmentation when it is too slow to run in real time as in Chapter 2, we propose a novel method to produce real-time performant segmentation networks. To this extend, we sacrifice the notion of universality described in Section 1.1.2 and develop a scene-specific method, named ARTHuS, that produces adaptive real-time match-specific networks for human segmentation in sports videos, without requiring any manual annotation.

This is achieved by a novel online knowledge distillation process, in which a fast student network is trained to mimic the output of an existing slow but effective universal teacher network, while being periodically updated to adjust to the latest play conditions. As a result, ARTHuS allows to build highly effective real-time human segmentation networks that evolve through the match and that sometimes outperform their teacher. The usefulness of producing adaptive match-specific networks and their excellent per-

formances are demonstrated quantitatively and qualitatively for soccer and basketball matches in this chapter.

This chapter is organized as follows. Section 3.1 recalls the challenges presented in Section 1.1.2 and particularizes them to the task of semantic segmentation. In Section 3.2, we describe the proposed method that solves these challenges in a general context. Then, Section 3.3 starts by presenting our specific setup along our evaluation methodology. Then, we provide an evaluation of the performances of the networks produced by ARTHuS quantitatively and qualitatively. These results demonstrate the superiority of adaptive match-specific networks over fixed sports-specific ones and even show that the students may outperform their teacher. Finally, we present a conclusion of this work in Section 3.4. Let us note that in Chapter 4, we extend this method to a multi-camera and multi-modal setup, showing the great generalization potential of online distillation.

PUBLICATION RELATED TO THIS CHAPTER

A. Cioppa, A. Deliège, M. Istasse, C. De Vleeschouwer, and M. Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2505–2514, Long Beach, California, USA, June 2019

Contributions. **(i)** We propose a novel method for human segmentation during live sports events. **(ii)** For a given match, our method produces an excellent segmentation network that evolves during the match, without having to manually annotate a single frame. **(iii)** We demonstrate the superiority of adaptive scene-specific networks over pre-trained ones. **(iv)** We show that the student may outperform sometimes its teacher in particular situations.

3.1 Introduction

Let us first recall that the task of semantic segmentation consists in assigning a label to each pixel of an image or a video. This provides rich information upon which an educated understanding of the whole content of the image can be drawn [71, 124]. Regarding sports videos, semantic segmentation could be on the basis of automatic systems for *e.g.*, tactics analysis, players interaction, event classification [37, 72], among numerous applications of computer vision in sports [134, 183]. However, despite being a valuable tool, semantic segmentation comes with various difficulties, which makes it an unsolved problem in the literature.

The first challenge is the annotation issue. Semantic segmentation is generally learned as a supervised task, hence requiring ground-truth pixel-wise annotations, a process that is too time-consuming to be handled manually in every new specific context, as evidenced by the absence of any such annotated dataset in sports. To

counterbalance the lack of annotated data, some authors generate synthetic training images, as in [62, 135, 148, 161], but the quality of the data generated is often difficult to assess [96]. Another possibility to circumvent the problem of annotations is to use transfer learning strategies, that is, models that have been trained on annotated datasets are reused in a novel environment, in which no (or few) annotation is performed [140, 201]. Nevertheless, current benchmark datasets do not cover every situation where semantic segmentation might be useful, which makes transfer learning effective only when the target domain is close to the source domain; the performances can rapidly decrease otherwise. A compromise can be found by using the transfer learning procedure on a subset of selected classes as in [22], such as humans in sports scenes, in order to have a partial semantic segmentation of the image.

Then comes the trade-off between speed and performance. For instance, on the Cityscapes dataset [39], the current best algorithms [32, 221] are rather slow, while the real-time ones [143, 203, 210, 219] are not as performant. Given that this dataset is meant to serve for the autonomous vehicles industry, it is essential that both performance-based and speed-based criteria are met simultaneously, which is not the case at the moment. These two aspects can also be required in sports video analysis to provide real-time accurate information about the ongoing match. A solution to benefit from the performances of a slow model (which can be designed as an ensemble of other models) and from the speed of a fast model is to perform a knowledge distillation from the slow one into the fast one [27, 84, 160, 204]. The slow accurate network has the role of a teacher, which is used as is to facilitate the training of the fast network, which has the role of a student that has to imitate its teacher's behavior on a same input dataset. After the training process, the student is supposed to be capable of real-time inference while showing good performances. In the case of semantic segmentation, this can alleviate the annotation problem for training the student if the teacher is considered reliable enough to provide approximations of unavailable ground-truth segmentation masks, which we denote as *surrogate ground truth*.

A last problem can be the lack of generalizability of the models, whose origin is at least twofold in sports video analysis: inter-sport variability, and intra-sport variability. It is currently too ambitious to hope for a universal system that can perform accurate semantic segmentation on any sports video, which underlines the need for developing sport-specific models. Besides, even within videos from a similar view of a single sport, some play conditions may change from one match to the next, such as the teams and the color of their outfits, the advertisements, the field, and some may even change during a match, such as weather conditions in the case of outdoor events. Fast algorithms can be less robust to such variations, which might make them non-reusable from one match to the next. Rather than trying to unify all these conditions within a same network, it might be more appropriate to (re)train a scene-specific network for every match in order to adjust to the conditions of that match, in the same spirit as [23, 142]. This is the motivation behind online learning (*e.g.*, [165] and references therein), in which the model is continuously updated thanks to the availability of new information.

In this work, we propose a novel method named ARTHuS, which stands for adaptive real-time human segmentation, to resolve at once the three issues presented above for human segmentation during live sports events. For a given match, ARTHuS produces a network that evolves during the match, without having to manually annotate a single frame. This is achieved through an online distillation of a slow but well-performing teacher network into a fast student network capable of real-time inference, which thus becomes match-specific.

3.2 Online knowledge distillation

The core problem addressed in this chapter is the issue of performing an excellent real-time human segmentation in sports videos. The ideal solution would be to develop an algorithm which is well-performing, fast, and universal so that it can be used to analyze every new match, regardless of the sport. However, this last aspect is out of reach at the moment. Consequently, in order to ensure performance and speed, we sacrifice the generalizability requirement. In fact, we target the opposite: we design a method that produces match-specific networks running accurately and in real-time on the match that they are meant to analyze.

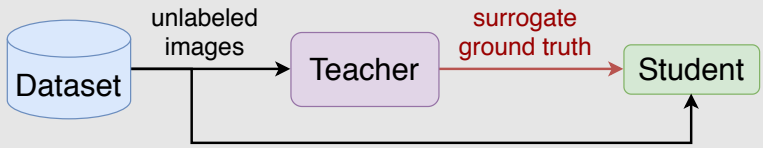
3.2.1 Elements borrowed from usual knowledge distillation

The first step consists in choosing a "universal" trained teacher network $\mathcal{T}$ and a student network $\mathcal{S}$, possibly untrained. The teacher $\mathcal{T}$ has to be as efficient as possible for our segmentation task, even if it means that its speed has been sacrificed for the sake of performance and universality. Therefore, the choice of this network can be entirely dictated by its performance on the targeted type of data, in our case, detecting soccer players. The student $\mathcal{S}$ has to be capable to perform the semantic segmentation task at least 25 frames per second to ensure real-time inference. Its choice is therefore only based on its computation time, which is entirely determined by its architecture and the server or embedded device on which it is meant to run. The student will then be trained on the surrogate ground truths produced by the teacher. At the end of the training process, the student network $\mathcal{S}$ has learned to mimic the behavior of $\mathcal{T}$ on $\mathcal{X}$ and hopefully has become good enough to serve as a real-time segmentation network for new unseen images. The distillation of $\mathcal{T}$ into S can thus be formulated as follows:

FORMULATION: (OFFLINE) KNOWLEDGE DISTILLATION

Given an unlabeled set of images $\mathcal{X}$, the (offline) knowledge distillation from $\mathcal{T}$ into $\mathcal{S}$ on $\mathcal{X}$ can be divided into two parts:

1. Compute $\mathcal{T}(\mathcal{X})$ by feeding every image of $\mathcal{X}$ into $\mathcal{T}$ to obtain surrogate ground-truth segmentation masks.
2. Learn $\mathcal{S}$ by supervised training with the dataset $\mathcal{D} = (\mathcal{X}, \mathcal{T}(\mathcal{X}))$.



It is important to note that this formulation of knowledge distillation is only a subset among the different possibilities of distillation that exist. In fact, knowledge distillation can also be useful when ground-truth data is available for the teacher and the student. Rather than producing surrogate ground truth, the objective of the teacher is now to produce some intermediate representation of the data, via its feature maps, and to mimic this representation of the data inside of the student network [184, 212]. This ensures that the student searches for the same kind of features as the teacher which hopefully lead to the same kind of performance. Of course, this is not the focus of this work since ground truth data is not available in our case and that we do not limit our method to student and teacher networks sharing the same kind of architecture.

3.2.2 Our online knowledge distillation method: ARTHuS

In the case of sports events, many factors may change from one match to the next or even within a single match. Therefore, there is no guarantee that $\mathcal{S}$, obtained by offline distillation, is able to generalize properly in novel circumstances, which motivates the idea to train a new student adaptively during every match. This leads us to propose a novel online distillation strategy which is composed of the following components:

COMPONENTS: ONLINE KNOWLEDGE DISTILLATION

1. A trained teacher $\mathcal{T}$ which remains fixed throughout the process and produces new surrogate ground-truth segmentation masks on the fly.
2. A student $\mathcal{S}^{\text{seg}}$ that performs a real-time segmentation of all the frames of the video stream. It adapts to the match as its weights are periodically updated; its k-th instance is noted $\mathcal{S}^{\text{seg}}_k$, where $\mathcal{S}^{\text{seg}}_0$ is its initial instance, used to segment the first frames of the stream.
3. A training dataset $\mathcal{D}$ that is updated during the match and whose k-th instance is $\mathcal{D}_k = (\mathcal{X}_k, \mathcal{T}(\mathcal{X}_k))$.
4. A duplicate of $\mathcal{S}^{\text{seg}}$, denoted $\mathcal{S}^{\text{train}}$, which is trained continuously during the match with the successive instances $\mathcal{D}_k$ of $\mathcal{D}$. This network is initialized with the weights of $\mathcal{S}^{\text{seg}}_0$ once at the beginning of the video, then its weights are gradually updated at each epoch with backpropagation.

The idea, which is represented in Figure 3.1 is the following. Given a live soccer game, we want to train a real-time student network, $\mathcal{S}^{\text{seg}}$, during live time for the task of semantic segmentation in a scene-specific approach. To do so, we use a universal slow teacher network, $\mathcal{T}$, to produce surrogate ground-truth annotations at a slower frame rate during the live soccer game. These surrogate ground truths are saved in an online dataset $\mathcal{D}$ to train a copy of the student network $\mathcal{S}^{\text{train}}$. Periodically, the weights of $\mathcal{S}^{\text{train}}$ are transferred to $\mathcal{S}^{\text{seg}}$, which thus becomes better at segmenting the players on this particular live game. All these processes take place in parallel during the live event and ensure a real-time accurate semantic segmentation of each frame.

Formally, our online distillation process can be explained in two recursive steps. Firstly, $\mathcal{S}^{\text{train}}$ starts training with $\mathcal{D}_k$ (*i.e.* $\mathcal{T}$ is distilled into $\mathcal{S}^{\text{train}}$ on $\mathcal{D}_k$) while $\mathcal{S}^{\text{seg}}_{k-1}$ is used to segment all the incoming frames and $\mathcal{T}(\mathcal{I})$ is computed for some subset $\mathcal{I}$ of these incoming frames. Secondly, after a predefined number of training epochs of $\mathcal{S}^{\text{train}}$ on $\mathcal{D}_k$, the weights of $\mathcal{S}^{\text{train}}$ are copied into $\mathcal{S}^{\text{seg}}_{k-1}$, which is thus updated into $\mathcal{S}^{\text{seg}}_k$, and $\mathcal{D}_k$ is updated into $\mathcal{D}_{k+1}$ as the newly available pairs $(\mathcal{I}, \mathcal{T}(\mathcal{I}))$ replace as many existing pairs of $\mathcal{D}_k$. After these updates, $\mathcal{D}_{k+1}$ is available, and $\mathcal{S}^{\text{train}}$ resumes its training but with $\mathcal{D}_{k+1}$, while the rest of the process follows. This way, $\mathcal{S}^{\text{seg}}$ is a real-time segmentation network which is constantly adjusted with respect to the latest play conditions, and therefore becomes specialized on the particular data distribution of the current game. The method is summarized in Algorithm 1 and the practical details are presented in Section 3.3.1.

As we focus on humans, through its successive instances $\mathcal{S}^{\text{seg}}_k$ ($k = 0, 1, \ldots$), $\mathcal{S}^{\text{seg}}$ can be seen as an adaptive real-time human segmentation network produced by our method, which we name ARTHuS, illustrated in Figure 3.2.

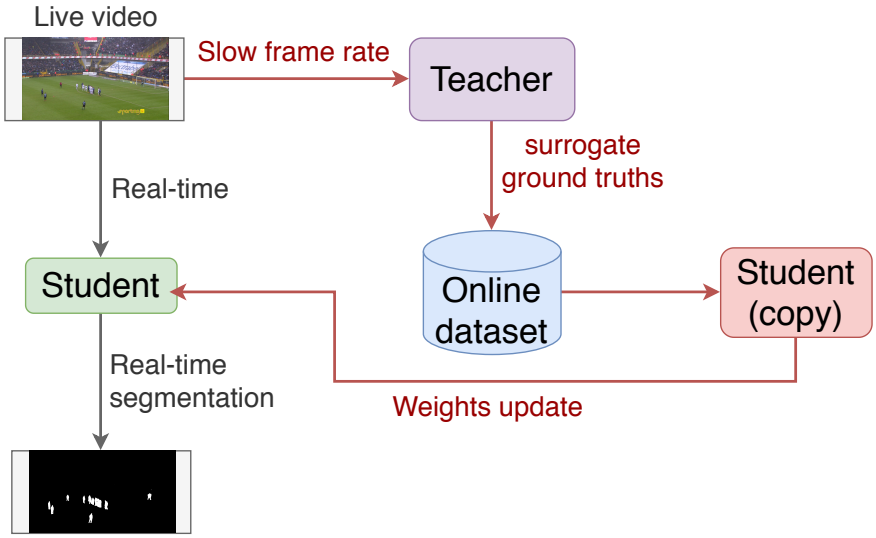


Figure 3.1: Overview of online distillation. Interactions between the different components of our online distillation process. The student network is periodically updated thanks to surrogate ground-truth annotations provided by the teacher, at a slower frame rate.

Algorithm 1 : The proposed online distillation algorithm.

Choose $\mathcal{T}$, initialize $\mathcal{S}_0^{\text{seg}}$ and $\mathcal{S}^{\text{train}}$, collect $\mathcal{D}_1$
while incoming video stream do
  while $\mathcal{S}^{\text{train}}$ trains with $\mathcal{D}_k$ do
    Segment all incoming frames with $\mathcal{S}_{k-1}^{\text{seg}}$
    Compute $\mathcal{T}(\mathcal{I})$ for some incoming frames $\mathcal{I}$
  end while
  $\mathcal{S}_{k-1}^{\text{seg}}$ becomes $\mathcal{S}_k^{\text{seg}}$ by copying weights of $\mathcal{S}^{\text{train}}$ into $\mathcal{S}_k^{\text{seg}}$
  $\mathcal{D}_k$ becomes $\mathcal{D}_{k+1}$ by replacing some data with $(\mathcal{I}, \mathcal{T}(\mathcal{I}))$
  Increment k by 1
end while

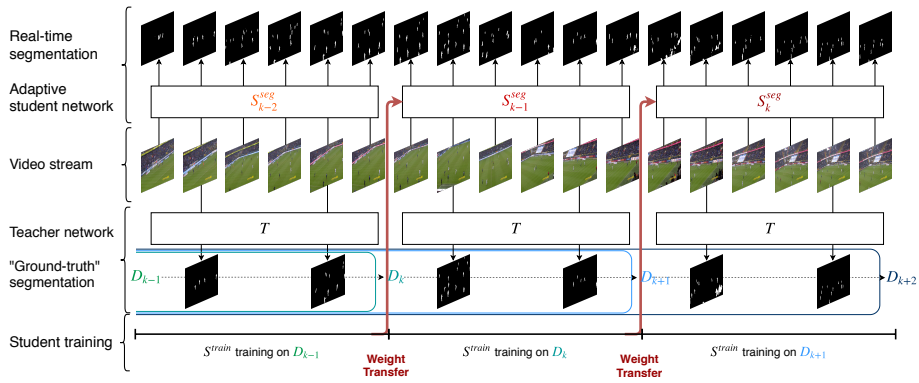


Figure 3.2: Online distillation pipeline. A real-time student segmentation network $\mathcal{S}^{\text{seg}}$ segments each frame of the video stream while its duplicate $\mathcal{S}^{\text{train}}$ continuously trains to mimic a slow but effective teacher segmentation network $\mathcal{T}$. The weights of $\mathcal{S}^{\text{train}}$ are periodically copied into $\mathcal{S}^{\text{seg}}$, which is thus consistently adapted to the latest match conditions and becomes match-specific.

3.3 Experiments

Our experiments are conducted on soccer and basketball videos. The objective is to segment all players and referees on the field, so excluding the public and other staff members that might be outside of the field. In this section, we first specify the particular settings in which we evaluate the performances (Section 3.3.1) and our evaluation methodology (3.3.2). The performances of the networks produced by ARTHuS are assessed quantitatively as described below and qualitatively through visual inspection of the results (Section 3.3.3). Finally, we look at some particular cases, where the student is actually better at segmenting the players than the teacher (Section 3.3.4). Let us note that additional results and detailed descriptions of the dataset and the training process can be found in Appendix B.

3.3.1 Specific settings for this work

ARTHuS involves several choices, such as the networks and the update strategies. In this work, some particular choices have been made and are described below, but it is important to underline that our method is not limited to these choices. Many variants can be derived from the founding principle described in Section 3.2 which will be further investigated in Chapter 4.

Data and hardware. The sports videos used in this work are composed of frames with dimensions 1920 × 1080 pixels from the main camera (see supplementary material for details). The framerates provided for the algorithms are reported for images of these

dimensions on one NVIDIA Tesla V100 GPU.

Teacher network. Our choice of a fixed well-performing universal teacher network $\mathcal{T}$ is Mask R-CNN [78], which runs at ≈ 2 fps on our images and has 33.8 million parameters. We use the PyTorch implementation available at facebookresearch/maskrcnn-benchmark. Mask R-CNN operates in two steps: a detection of regions of interest followed by a segmentation within these regions. This network outputs several bounding boxes predictions with corresponding labels and segmentation masks inside the boxes. In order to select only interesting humans for our online training process, we keep the segmentation masks provided inside the boxes whose label corresponds to "human". In order to focus only on players present on the field, after the detection step of Mask R-CNN, only the regions that intersect the field are kept. This filtering is performed using the segmentation mask of the field, which we compute as further discussed in Chapter 5 for our soccer experiments, and which is provided in a calibration file with the data for our basketball experiments. The whole process of collecting the results of Mask R-CNN and refining them to keep humans on the field produces training images for the instances of $\mathcal{D}$ at the speed of ≈ 1 fps.

Student network. We choose one of our own architecture also used in a further chapter of this thesis (Chapter 5), named TinyNet and completely described in Section A.1, for the fast student network $\mathcal{S}^{\text{seg}}$. It is a lightweight variant of PSPNet [221] with only 0.6 million parameters, which is about 100 times less than the original PSPNet. Its inference speed is about 0.0165 seconds per image (≈ 60 fps) and the training time of its duplicate $\mathcal{S}^{\text{train}}$ is ≈ 0.08 second per image.

Initialization. At the beginning of a new video stream, $\mathcal{D}_0$ is empty and the first minutes are used to collect and annotate data with $\mathcal{T}$ in order to build $\mathcal{D}_1$, the first non-empty instance of $\mathcal{D}$. During that time, $\mathcal{S}^{\text{train}}$ is on stand-by until $\mathcal{D}_1$ contains enough images to start its training. We consider that $\mathcal{D}_1$ is complete when it is composed of 200 annotated frames. Regarding $\mathcal{S}_0^{\text{seg}}$ (the first instance of $\mathcal{S}^{\text{seg}}$) that segments all the frames of the video stream during the building of $\mathcal{D}_1$, two approaches are tested: a random initialization, and a copy of a network pre-trained by usual offline distillation of $\mathcal{T}$ on six other matches of the same sport (see Appendix B), which is noted $\mathcal{S}_{\text{pretrained}}$.

Training. In our setting, each $\mathcal{D}_k$ is composed of 200 frames but $\mathcal{S}^{\text{train}}$ is actually trained on a subset of $\mathcal{D}_k$ covering the same game duration, built by selecting one frame every three frames. This subsampling is performed to speed up the training process of $\mathcal{S}^{\text{train}}$ and thus increases the frequency of the updates of $\mathcal{S}^{\text{seg}}$, which strengthens its adaptability during the match. We choose to train $\mathcal{S}^{\text{train}}$ during 1 epoch with the subsampled version of $\mathcal{D}_k$ before updating $\mathcal{S}_{k-1}^{\text{seg}}$ into $\mathcal{S}_k^{\text{seg}}$ and $\mathcal{D}_k$ into $\mathcal{D}_{k+1}$. It is trained one image at a time (no batches) using the Adam optimizer [110], and takes approximately $200/3 \times 0.08 = 5.3$ seconds per epoch. The weighted cross-entropy loss is used to handle, to some extent, the imbalance between human pixels and background pixels, whose ratio ranges from 1/50 to 1/20 in our images. The weighting factor of the loss is recomputed for each $\mathcal{D}_k$.

Updating $\mathcal{D}_k$. For each $k \geq 1$, the update strategy of $\mathcal{D}_k$ follows the "first in, first out" rule, that is, the oldest training images are replaced by the new ones. We choose to replace the oldest frames instead of just adding the new ones in order to ensure that $\mathcal{S}_k^{\text{seg}}$ is adapted to the latest match conditions and to keep the size of $\mathcal{D}_k$ constant, which allows to have an almost constant training time per epoch for $\mathcal{S}^{\text{train}}$ and thus regular updates for $\mathcal{S}^{\text{seg}}$.

3.3.2 Evaluation methodology

As each instance $\mathcal{S}_k^{\text{seg}}(k = 0, 1, \ldots)$ of $\mathcal{S}^{\text{seg}}$ produces binary masks indicating whether the pixels belong to a human (output = 1) or not (output = 0), performance metrics derived from confusion matrices can be used to represent its performances, provided that ground-truth masks are available. In such a case, the F_1 score of $\mathcal{S}_k^{\text{seg}}$ is a well-suited metric; it is computed as

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.1}$$

where TP denotes the number of true positives (pixels correctly predicted as humans), FP the number of false positives (pixels erroneously predicted as humans), and FN the number of false negatives (pixels erroneously predicted as non-humans). However, in our case, it is difficult to obtain a large amount of ground-truth masks. For this reason, we perform the evaluation of $\mathcal{S}_k^{\text{seg}}$ in two steps. First, in Section 3.3.3, the evaluation is computed on frames that are annotated by $\mathcal{T}$, which we consider as sufficiently good approximations of the unavailable ground-truth masks that can be used as references. A large number of these annotated frames is available and this evaluation is meant to provide a first overview of the performances of $\mathcal{S}_k^{\text{seg}}$. Then, in Section 3.3.4, we manually correct the annotations of a subset of these frames to build a cleaner test dataset. This helps us showing that $\mathcal{S}_k^{\text{seg}}$ is mostly correct even when $\mathcal{T}$ fails. It also allows to support the previous results and it attests of the reliability of that evaluation technique in our context.

As $\mathcal{S}_k^{\text{seg}}$ can be regarded as a network trained on the frames annotated by $\mathcal{T}$ that compose $\mathcal{D}_1, ..., \mathcal{D}_k$, its evaluation has to be conducted a posteriori (not in real time) on frames recorded after those present in $\mathcal{D}_k$. We choose to constitute the test set of $\mathcal{S}_k^{\text{seg}}$ with the N frames annotated by $\mathcal{T}$ following those of $\mathcal{D}_k$ and used to build the next instances of $\mathcal{D}$. In this work, we set $N = 300$, which spans the next five minutes of video given the framerate of $\mathcal{T}$. This way, we compute the F_1 score of each $\mathcal{S}_k^{\text{seg}}$ on its test set by accumulating the confusion matrices of each frame of the set according to the principle of summarization explained in [144]. This gives the temporal evolution of the performances of $\mathcal{S}^{\text{seg}}$ throughout the video.

In order to handle the possible uncertainty of $\mathcal{T}$ at the borders of the humans to segment, which represents the intrinsic difficulty to perform pixel-wise annotations, some margins are drawn outside and inside these borders, whose pixels are excluded from

the computation of F_1 scores. Technically speaking, these margins are computed as the Beucher gradient of the masks with a centered 7×7 structuring element, and correspond to the set difference between the morphological dilation and erosion. This practice is common in domains such as background subtraction [199], which is close to our problem in terms of evaluation of performances. This is illustrated in Figure 3.3.

Figure 3.3: Evaluation borders. The Beucher gradients of the masks produced by $\mathcal{T}$ define thin margins (in red) whose pixels are excluded from the quantitative evaluation process in order to reduce the impact of the lack of accuracy of $\mathcal{T}$ at the borders of the masks on the evaluation.

3.3.3 Performances of our method

We assess the performances of the networks produced by ARTHuS on two test matches: one for soccer, one for basketball. The soccer match is the 2013 Belgian Jupiler Pro League match between FC Bruges and Anderlecht. The basketball match is the 2019 French Jeep Elite League match between Cholet and Boulazac. We chose these two matches because they both contain one unusual event out of actual game time (involving mascots), to demonstrate that our method can quickly recover from perturbations (more details are provided in Appendix C).

For each of them, we test the two strategies mentioned in Section 3.2 for the initialization of $\mathcal{S}_0^{\text{seg}}$: random initialization for training it online from scratch, and training it online from a sports-specific version that has been pre-trained by regular offline distillation of $\mathcal{T}$ on six other matches of the same sport, which we note $\mathcal{S}_{\text{pretrained}}$. We also compare these two approaches with the performances of $\mathcal{S}_{\text{pretrained}}$ when it is kept as is

throughout the test, in order to assess its generalization capabilities and to illustrate the interest of producing adaptive match-specific networks.

The evolution of the F_1 score (with respect to the masks provided by $\mathcal{T}$) of each experiment can be found in Figure 3.4, where the unusual game event is marked out. It can be inferred that ARTHuS works well in practice, as indicated by the high level of performance achieved by the networks produced, regardless of the initialization strategy or the sport. In particular, even though the networks $\mathcal{S}_{\text{pretrained}}$ already have good generalization skills, the networks that are trained adaptively to become match-specific always achieve better performances after a few minutes of match. Even the networks trained from scratch with $\mathcal{S}_0^{\text{seg}}$ randomly initialized, hence without any prior knowledge on what a human on a soccer or basketball field is, eventually outperform $\mathcal{S}_{\text{pretrained}}$ and come close to those that are re-trained from $\mathcal{S}_{\text{pretrained}}$. Furthermore, the networks trained online quickly recover to excellent performances after the unusual game event. Overall, the best performances are obtained with the adaptive networks that are initialized as $\mathcal{S}_{\text{pretrained}}$. These observations validate the effectiveness of the method and strengthen our point that producing adaptive match-specific networks leads to better results than using fixed sports-specific networks.

The need for match-specific networks is reinforced by the following elements. Regarding the soccer experiment, $\mathcal{S}_{\text{pretrained}}$ has been trained on matches from the UEFA Euro 2016. When we tested $\mathcal{S}_{\text{pretrained}}$ on another match from that competition, a smaller gain in performance was noted when retraining it online. This was presumably because the advertisements, camera views, stadiums, and lighting conditions, were similar to those already seen by $\mathcal{S}_{\text{pretrained}}$. However, the test match evaluated in Figure 3.4 is taken from another competition, the Belgian Jupiler Pro League. This match is thus rather different from those used to train $\mathcal{S}_{\text{pretrained}}$, which explains the large gap in performances between $\mathcal{S}_{\text{pretrained}}$ and $\mathcal{S}^{\text{seg}}$ on that match. Regarding the basketball experiment, the matches used to train $\mathcal{S}_{\text{pretrained}}$ belong to the same competition, but the stadiums are very different from one match to another. Therefore, there is no guarantee that $\mathcal{S}_{\text{pretrained}}$ is able to generalize correctly, and our experiment confirm that it is again beneficial to re-train it online to produce a match-specific network.

From a visual perspective, some results are displayed in Figure 3.5 for $\mathcal{S}_{\text{pretrained}}$ and for the networks that have been re-trained online with our method with $\mathcal{S}_{\text{pretrained}}$ as initialization. The differences in the performances reported in Figure 3.4 are backed up by Figure 3.5. As $\mathcal{S}_{\text{pretrained}}$ is not specific to the test match, it cannot handle some of its peculiarities. As a result, it produces more false positives, such as the lines of the new soccer field or elements of the new basketball stadium, and more false negatives, such as partially unsegmented players, which are correctly classified by the instances of $\mathcal{S}^{\text{seg}}$ in use when these frames were recorded.

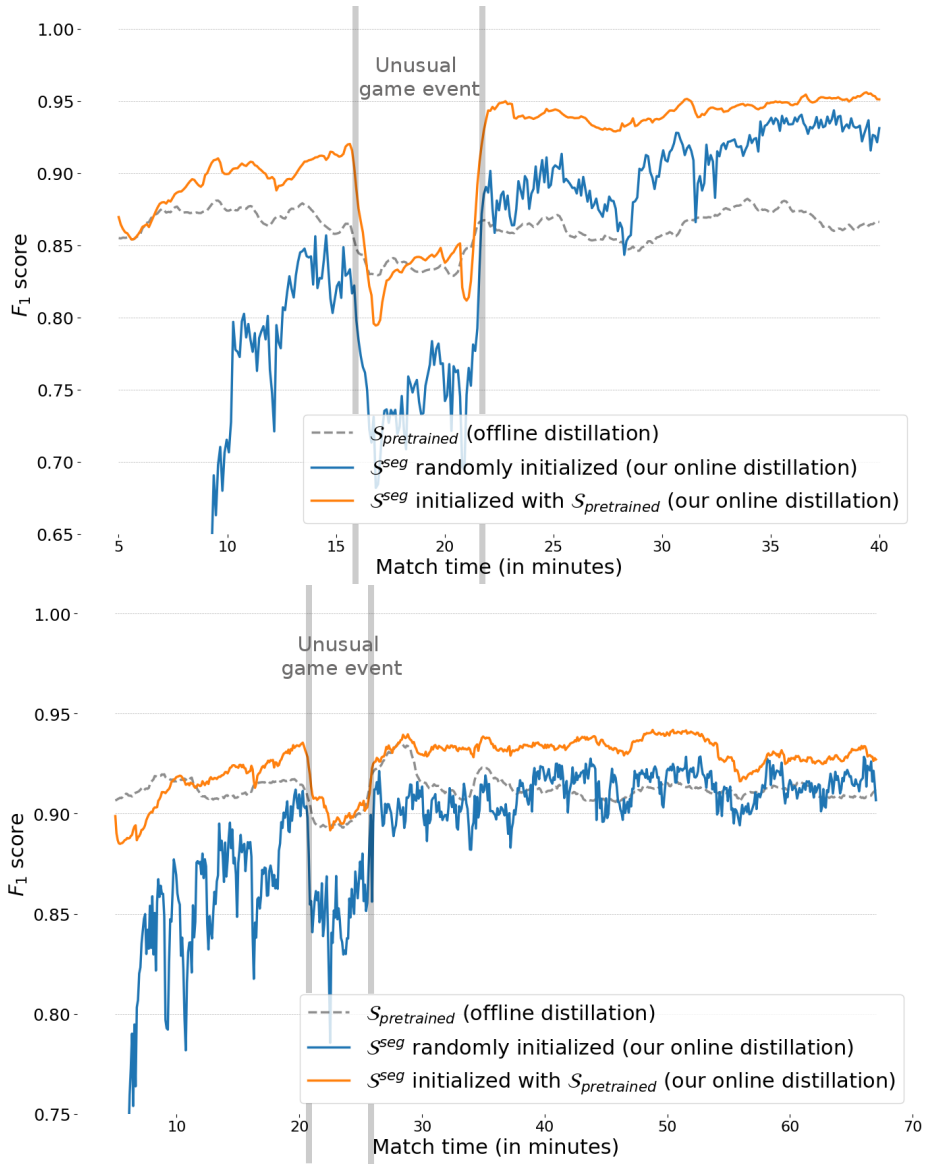


Figure 3.4: Performances of online distillation. Evolution of the performances of several variants of our distilled models through their F_1 score computed with respect to the masks provided by $\mathcal{T}$ for the soccer (top) and basketball (bottom) test matches.

Figure 3.5: Qualitative results. Human segmentation results produced by $\mathcal{S}_{\text{pretrained}}$ (left column) and by our adaptive match-specific network $\mathcal{S}^{\text{seg}}$ produced by ARTHuS (right column) initialized with $\mathcal{S}_{\text{pretrained}}$. The effectiveness and usefulness of the adaptive network $\mathcal{S}^{\text{seg}}$ can be observed.

3.3.4 Does the student outperform its teacher?

A question that arises with knowledge distillation is whether the student network outperforms its teacher, which may occur in practice [63, 160, 209]. In our case, this question is further motivated by the observation that $\mathcal{T}$ sometimes makes mistakes that $\mathcal{S}^{\text{seg}}$ does not, as illustrated in Figure 3.6. This qualitative inspection suggests that $\mathcal{S}^{\text{seg}}$ may indeed outperform $\mathcal{T}$, depending on the viewer's subjective expectations to consider that $\mathcal{S}^{\text{seg}}$ surpasses $\mathcal{T}$.

From a quantitative point of view, the evaluation method presented above cannot help answering that question since the output of $\mathcal{T}$ is considered to be the ground truth with respect to which the performances of the instances $\mathcal{S}^{\text{seg}}_k$ ($k = 0, 1, \ldots$) of $\mathcal{S}^{\text{seg}}$ are necessarily inferior or, at most, equal. Besides, the curves presented in Figure 3.4 are slightly flawed by the mistakes of $\mathcal{T}$ and require a manual investigation to be corrected. Manually annotating all the test frames would be too time-consuming. Also, it can be noted that $\mathcal{T}$ already segments almost perfectly most of the humans present in the test videos and that manual annotations would not be better in many cases. Therefore, we propose to build a semi-manually annotated test set, in which we manually correct the segmentation masks provided by $\mathcal{T}$ by either removing non-human pixels from the masks or by adding missed human pixels in the masks. This procedure is performed for a subset of the test frames because of limited annotation resources. In the soccer (resp. basketball) case, 3.5% (resp. 2.5%) of the annotations have been modified, which indicates that $\mathcal{T}$ is reliable most of the time.

Considering that these corrected frames constitute the real ground truth, we can re-evaluate the performances of $\mathcal{S}^{\text{seg}}_k$ ($k = 0, 1, \ldots$) through the match. As it can be seen in Figure 3.7, the F_1 score increases by a comfortable margin compared with the previous evaluation, which confirms the intuition that, when $\mathcal{T}$ is wrong, $\mathcal{S}^{\text{seg}}$ is actually mostly right. To further support this claim, we also compute the performances that $\mathcal{S}^{\text{seg}}$ would achieve if we assume that it makes no mistakes on the corrected pixels (those where $\mathcal{T}$ was considered to be wrong). This curve is also represented in Figure 3.7. This way, we can better quantify how good $\mathcal{S}^{\text{seg}}$ is on these new annotations, and it turns out that it is almost perfect since its adjusted performance curves are close to their upper bounds. Given that most of the annotations are still those from $\mathcal{T}$, the performance curves of $\mathcal{T}$ are unfairly higher and hence are not plotted for the sake of clarity. Let us note that the similarity between the shapes of the initial curves and the corrected curves indicates that the first approach gives a valid overview of the evolution of the performances of the networks.

Even though it is difficult to decide whether $\mathcal{S}^{\text{seg}}$ outperforms $\mathcal{T}$ or not, several qualitative and quantitative experiments show that the gap between the two is negligible and that $\mathcal{S}^{\text{seg}}$ is at least nearly as good as $\mathcal{T}$, if not slightly better.

Figure 3.6: Qualitative comparison between the student and the teacher. Example of a case where the teacher (top) does not provide a reliable output, while the instance $\mathcal{S}_k^{\text{seg}}$ (bottom) of the student network $\mathcal{S}^{\text{seg}}$ in use for this frame is actually almost flawless.

3.4 Conclusion

In this chapter, we proposed a novel method, named ARTHuS, that produces adaptive real-time human segmentation networks without requiring manual annotations. It is based on a novel online knowledge distillation, in which a fast student network is trained adaptively with data annotated by a slow pre-trained teacher. We demonstrate the effectiveness of our method quantitatively and qualitatively on soccer and basketball matches. We show that match-specific networks outperform fixed pre-trained sports-specific networks, and that they eventually outperform their teacher on some occasions. We also show that ARTHuS works well with the choice of a simple lightweight student networks.

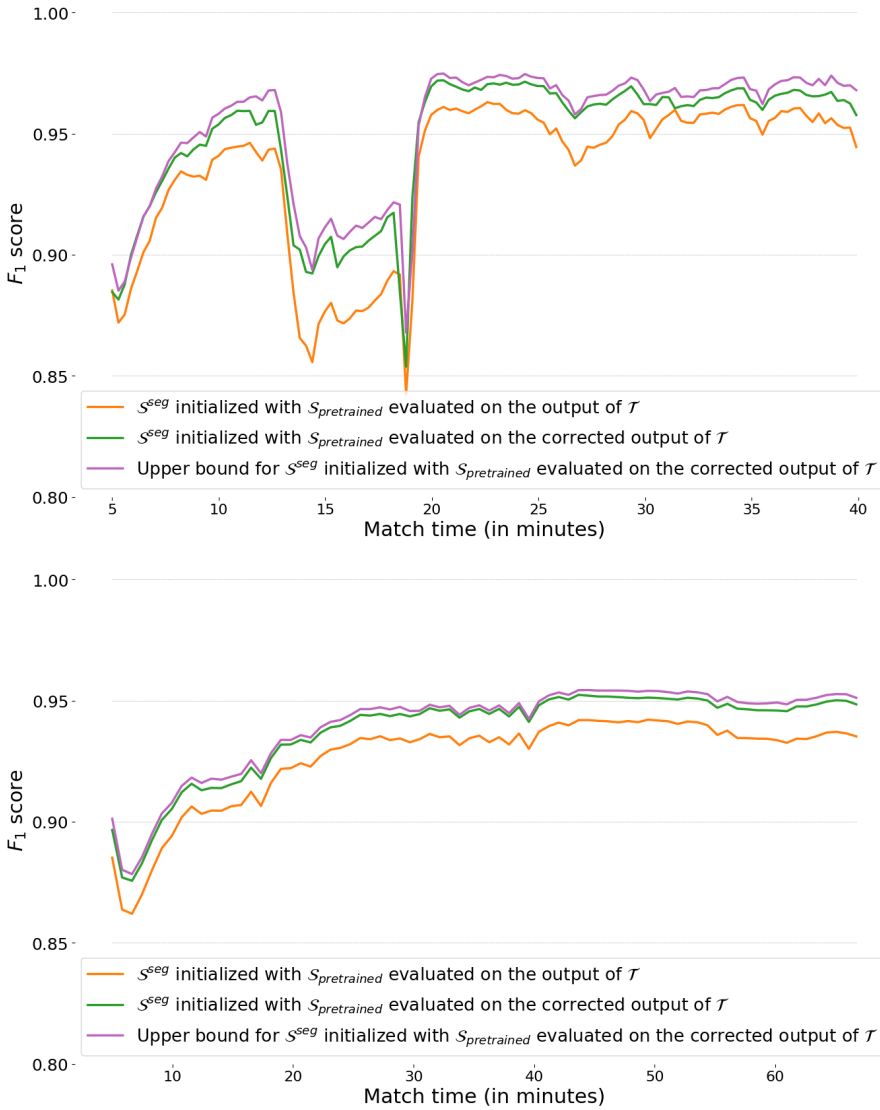


Figure 3.7: Quantitative comparison between the student and the teacher. The previous curves (orange) are adjusted (green) by evaluating the instances $\mathcal{S}^{\text{seg}}_k$ $(k = 0, 1, \ldots)$ of $\mathcal{S}^{\text{seg}}$ on the manually corrected test frames for the soccer (top) and basketball (bottom) matches. The green curves are higher, which suggests that $\mathcal{S}^{\text{seg}}$ is right when $\mathcal{T}$ is wrong. The maximum performances that $\mathcal{S}^{\text{seg}}$ would achieve if we suppose that $\mathcal{S}^{\text{seg}}$ is correct when $\mathcal{T}$ is wrong are plotted in purple. Since the green and the purple curves are close, $\mathcal{S}^{\text{seg}}$ is almost perfect on the pixels mislabeled by $\mathcal{T}$.

Although ARTHuS provides promising results, there is still room for improvement that can be investigated in future works. For instance, the update strategy of $\mathcal{D}$ can be revised in order to keep the possibility to use older frames if they are more informative than the new ones. We could also use an extra dataset that remains fixed and that would be composed of annotated frames of other matches, in order to ensure minimal generalization capabilities and enhance the robustness to possible anomalous events in the ongoing match. Besides, we can leverage the segmentation skills of our method to perform further analyses in order to develop real-time scene understanding techniques. However, we choose to mainly focus on generalizing the method itself to other tasks and types of data rather than fine-tuning its parameters for the particular case of semantic segmentation. This is the subject of the next chapter.

CHAPTER 4

A multi-modal and multi-view extension to our online distillation method

Contents

In this chapter, we provide an extension of the online distillation method presented in Chapter 3 in a real-world application which consists in monitoring the occupancy of public sports facilities. The use of online distillation is motivated by the fact that this system needs to be installed in many stadiums with different configurations. Therefore, an adaptive method such as the one of Chapter 3 allows to avoid training a model specific to each stadium prior to its installation, which is a huge economic advantage. In this way, the system can simply be placed in any stadium and initialized from scratch, as online distillation will train the network rapidly on this particular stadium and achieve great performances.

In the case of a soccer field, the area to cover is large, thus several regular cameras should be used to cover the whole field, which would makes the setup expensive and complex. As an alternative, we develop a method that detects players from a unique cheap and wide-angle fisheye camera assisted by a single narrow-angle thermal camera.

The fisheye camera provides a single view of the whole field, which is convenient to detect all players with a student network. The thermal camera provides a reliable way to detect the players since they have a heat body temperature very different from the environment, which makes it a great source of information for the teacher.

One major challenge is that we need to train a network in a knowledge distillation approach in which the student and the teacher have different modalities and a different view of the same scene. To do so, we design a custom data augmentation combined with a motion detection algorithm to handle the training in the region of the fisheye camera not covered by the thermal one. We show that our solution is effective in detecting players on the whole field filmed by the fisheye camera. We evaluate it quantitatively and qualitatively in the case of an online distillation, where the student detects players in real time while being continuously adapted to the latest video conditions.

This chapter is organized as follows. In Chapter 4.1, we motivate the choice of our camera setup and the need for online distillation and describe our data in Chapter 4.2. Then, in Chapter 4.3, we present our adaptation of online distillation in the case of our multi-modal and multi-view case. Descriptions of the experiments and their results are provided in Chapter 4.4. Finally, in Chapter 4.5, we draw some conclusions. on this work as well as online distillation.

PUBLICATION RELATED TO THIS CHAPTER

A. Cioppa, A. Deliège, N. Ul Huda, R. Gade, M. Van Droogenbroeck, and T. Moeslund. Multimodal and multiview distillation for real-time player detection on a football field. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 3846–3855, Seattle, Washington, USA, June 2020

Contributions. **(i)** We provide a generalization of online distillation in a multi-modal and multi-camera setup. **(ii)** We show how two different image modalities and fields of view can be combined in a student-teacher distillation approach. **(iii)** We show how a student network can be trained to detect players outside the field of view of the teacher, through a combination of a custom data augmentation process and a motion detection algorithm.

4.1 Introduction

Local sports fields can be expensive to construct and maintain, especially those built with artificial turf. Therefore, it is important to monitor and then optimize the occupancy of existing fields and stadiums. Furthermore, an automatic occupancy analysis method may open up new possibilities within real-time information and booking. In this Chapter, we propose a robust and cost-effective method for player detection and counting in a soccer field that relies on the concept of online distillation introduced in Chapter 3.

For robust video monitoring of outdoor soccer fields, one main challenge is the size of the field. A field may be covered by either several regular cameras, which makes the setup rather complex and expensive, or it is possible to use a camera with a wide field of view, such as a fisheye camera. However, with a fisheye camera covering the entire soccer field, the players will appear small and have different orientation in the image due to the lens distortion. Player detection on these types of images is therefore not a trivial task. Another main challenge in outdoor environments is varying lighting conditions. Even though a soccer field may be illuminated during nights, lighting conditions will change during the day due to changing weather, position of the sun, and the effect of artificial lighting. To avoid problems with difficult lighting conditions, thermal cameras may be considered. These cameras capture only thermal infrared radiation, which represents the temperature in the scene, hence they are more independent of lighting and normally eases the task of person detection because people have a temperature different from the background [66]. However, thermal cameras are expensive and due to their limited field of view and resolution, several cameras would be needed to cover a soccer field.

To construct a camera setup that is reasonable in price level and at the same time robust to changes in weather and lighting conditions, we propose to use one fisheye RGB and one thermal camera co-located at the side of the field. An illustration of the setup and example images from the two cameras are shown in Figure 4.1. Only the fisheye camera will cover the entire field, while the detections obtained directly from the thermal camera will serve to provide some surrogate ground truths for teaching a network.

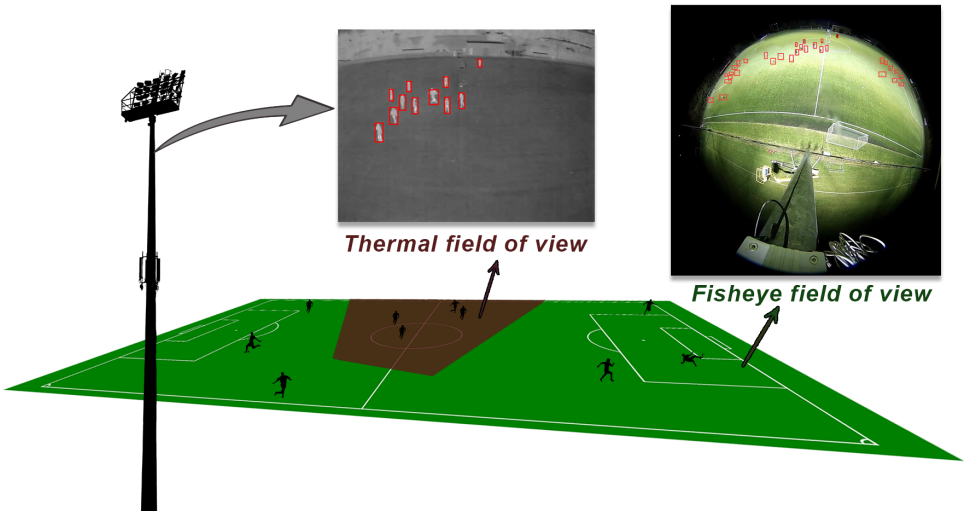


Figure 4.1: Illustration of the problem handled in this chapter. We leverage the detections made on a thermal image on a part of the field to detect all the players on the whole field on the fisheye image.

In the process of engineering a solution, we focus on an additional type of low-level semantics, called *object detection*. This task consists in regressing bounding boxes around the object of a scene. It is a very popular task these days, supported by many datasets such as COCO [123] or PASCAL_VOC [55] on which state-of-the-art methods such as Yolo [156] and Mask-RCNN [78] compete. Unlike semantic segmentation, object detection allows to differentiate the instances of a class in a scene, but does not provide information about the true boundaries of the objects. Therefore, semantic segmentation and object detection are often seen as complementary semantics about a scene, and have been recently merged into a single task called panoptic segmentation [112].

Detection of players in sports fields is the first step of vision systems for sports applications, like occupancy analysis, tracking, performance analysis, etc. [183]. Background subtraction based methods have often been used for player detection due to the fast processing time that makes it well-suited for real-time applications. It has been applied for static cameras [7, 159] and for moving cameras in the case of uniformly colored surfaces [153]. However, noise should be expected due to, *e.g.*, other moving objects, similar colors in foreground and background, changing lighting conditions, and shadows. It has also been proposed to use classic person detection methods like using the AdaBoost algorithm for training a linear classifier with HOG features for detecting players in Australian Rules Football [57], or similarly with AdaBoost and Haar features for player detection in basketball [98] and baseball [130].

More recently, like for general object detection, CNN-based methods have also been the dominant trend for detecting sports players. In [128] a shallow CNN was trained to detect players on a hockey field, while others use pre-trained networks like Mask R-CNN for handball videos [149] and basketball videos [207], or YOLO for handball videos [28]. In [218], a reverse connected convolutional neural network (RC-CNN) is proposed for player detection. The reverse connected modules are embedded into the CNN to pass semantic information captured by deep layers back to shallower layers.

Fisheye cameras have been widely used for person detection because of their advantage of wide viewing angle. Methods using a single camera setup have been reported for surveillance [109], automobiles [115], indoor environment [166, 197] and outdoor sports field [89]. In these methods, the setup was used for pedestrian detection, tracking and occupancy analysis. Multiple camera setups are also proposed to detect persons for similar applications [14, 138, 198]. However, the main disadvantages with fisheye cameras are the distortion on the borders and the lower image quality in low lighting conditions.

Thermal cameras have long been used in practice because of their efficiency in bad lighting conditions. The range of applications varies from industrial uses to daily life traffic and surveillance [66]. Various methods based on thermal cameras have been proposed for person detection, such as feature extraction and threshold based methods [41, 64, 65, 217], HOG methods [117, 188], machine learning techniques [90] and deep neural networks [82, 83, 190]. A dataset and a trained network for people detection on outdoor thermal images have been proposed in [190]. The disadvantage of thermal

cameras is their expensive cost and their reduced field of view.

Our aim is to circumvent, by means of a CNN-based method for player detection, the limitations of both fisheye and thermal cameras, by combining these modalities and teach the network for the fisheye camera with detections from the thermal camera, in a student-teacher online distillation approach.

4.2 Data acquisition and calibration

The used data consist of video streams from two different cameras: a fisheye camera and a thermal camera. Both cameras are installed on the same pole at the side of a soccer field, as illustrated in Figure 4.1. The thermal camera is placed approximately 9.8 meters above the ground and the fisheye camera is installed at 9.5 meters. By doing so, the field of view of the fisheye camera covers the whole soccer field, whereas the thermal camera covers the central area, as shown in Figure 4.1. In this setup, the field of view of the thermal camera represents 6% of the fisheye image, and covers 22% of the soccer field as seen by the fisheye camera. Let us note that several teams use the field simultaneously for a training session during the video. Hence, the players are performing different activities, such as moving goals or performing various exercises. Therefore, the players can be found in different postures in any part of the field, which makes our setup even more challenging.

The fisheye video stream is recorded using a Hikvision Fisheye Network Camera with a resolution of 1280×1280 pixels and a field of view of $360°$. The thermal video stream is recorded using an Axis Q1922 camera that has a resolution of 640×480 pixels and $57°$ of horizontal viewing angle. The videos were recorded during one hour in an amateur soccer field in December 2017, at night time with artificial light illuminating the field. The fisheye camera records the video at 12 fps. The thermal camera initially records the video at 30 fps, which is then re-sampled at 12 fps to allow a synchronization of the two streams. A proper camera calibration and registration between fisheye and thermal images is required for the transferability of points of interest.

First, a calibration of the internal parameters of each camera is performed following the procedure described in [139]. For the thermal camera, an A3-sized 10 mm polystyrene foam board is used as backdrop and a board of the same size with cut-out squares is used as checkerboard. In order to obtain a suitable contrast, the backdrop is heated and the checkerboard is placed at room temperature before the calibration. For the fisheye camera calibration, a checkerboard of 25×25 centimeters is used. Finally, the camera parameters derived from the calibration are obtained with a Matlab toolbox [17].

Second, we perform the registration between the two cameras. We undistort the images of the cameras using the internal parameters obtained previously. We manually choose several points of interest on the undistorted soccer field to compute the homography between the cameras, following [131]. These points are player feet positions for the

players seen by the two cameras. The projection of the thermal image onto the fisheye image is shown in Figure 4.2.

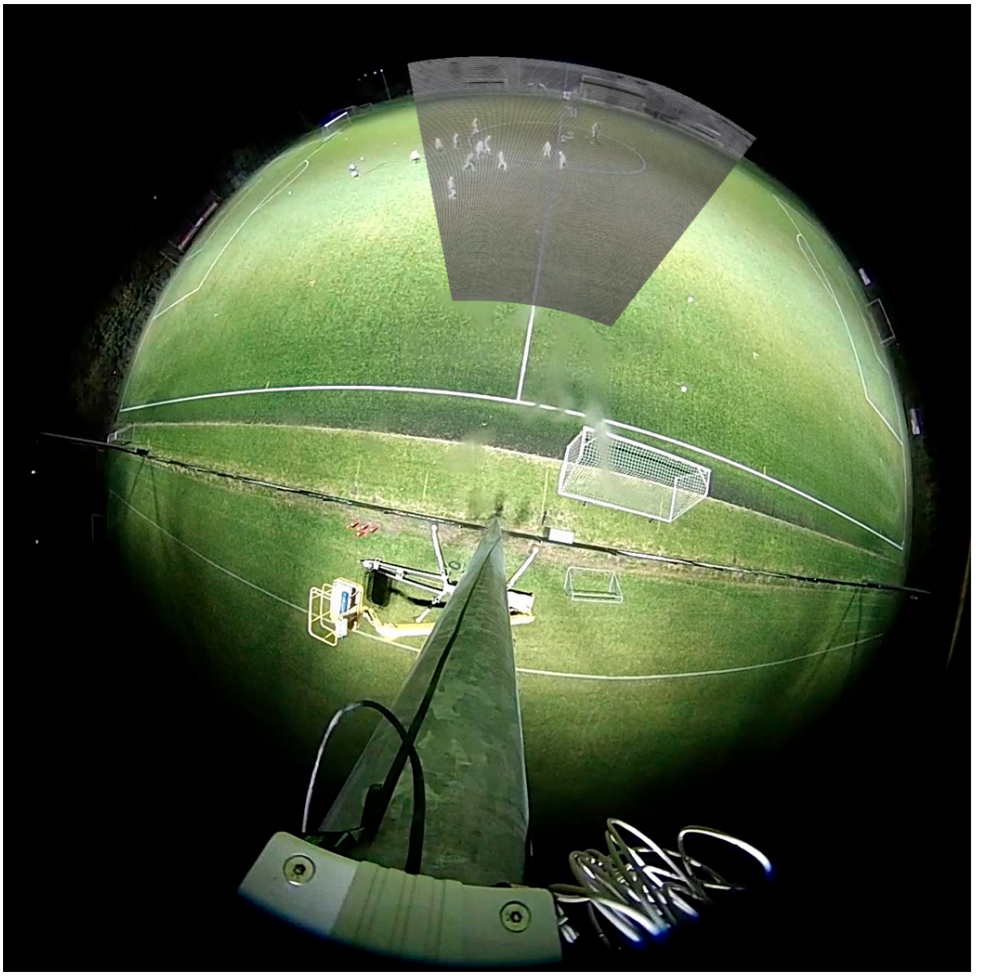


Figure 4.2: Projection of the thermal image onto the fisheye image. The thermal camera sees only ≈ 22% of the soccer field pixels of the fisheye image.

4.3 Multi-modal and multi-view online distillation

4.3.1 Formulation and notations

A general formulation of the problem tackled in this chapter is the following. Given a network performing a detection task on data from a camera, how can we train a real-time network for the same detection task on data from another camera with a possibly different modality and a different field of view of the same scene? In this section, we describe our solution for this problem in general terms, and we also explain how each step is particularized for our practical use case. Our use case consists in the task of player detection on a soccer field given a network able to detect players on a fixed thermal camera with a narrow field of view, which is used to train another detection network on data from a fixed fisheye camera with a wide field of view. This is illustrated in Figure 4.1.

We handle this problem with a teacher-student online distillation approach, in which the output of a trained teacher network $\mathcal{T}$ serves as surrogate ground truth to train a student network $\mathcal{S}$ (see [196] for a recent review of knowledge distillation methods). This is the method applied in Chapter 3 for segmenting soccer and basketball players in real time by distilling a slow $\mathcal{T}$ (Mask R-CNN [78]) into a fast $\mathcal{S}$ (TinyNet Chapter 5). However, in the previous chapter, $\mathcal{T}$ and $\mathcal{S}$ used the same video feed, which implies that $\mathcal{S}$ could be directly (no transformation needed) and entirely (no missing ground truth) supervised by $\mathcal{T}$.

Our setup is more challenging as $\mathcal{T}$ and $\mathcal{S}$ process the video feeds of two cameras $\mathcal{C}_\mathcal{T}$ and $\mathcal{C}_\mathcal{S}$ with different modalities and fields of view. Having different modalities prevents us from using $\mathcal{T}$ on the feed of $\mathcal{C}_\mathcal{S}$, and having different fields of view prevents us from directly and entirely supervising $\mathcal{S}$. We assume that $\mathcal{C}_\mathcal{T}$ and $\mathcal{C}_\mathcal{S}$ are synchronized, such that they capture frames $\mathcal{C}_\mathcal{T}(t)$ and $\mathcal{C}_\mathcal{S}(t)$ simultaneously at each capture time t. We also assume that the projection from $\mathcal{C}_\mathcal{T}(t)$ to $\mathcal{C}_\mathcal{S}(t)$, expressed in terms of pixel coordinates, is known from the preliminary calibration step explained in the previous section. We note $\mathbb{P}$ the area of $\mathcal{C}_\mathcal{S}(t)$ representing the projection on $\mathcal{C}_\mathcal{S}(t)$ of the part of the scene also filmed by $\mathcal{C}_\mathcal{T}$ (shown in Figure 4.3), also called the *common region*. The remaining part of $\mathcal{C}_\mathcal{S}(t)$ is filmed by $\mathcal{C}_\mathcal{S}$ only and is noted $\overline{\mathbb{P}}$. As both cameras are fixed, this partition of $\mathcal{C}_\mathcal{S}(t)$ is independent of t.

In order to train $\mathcal{S}$, we need surrogate ground-truth bounding boxes both in $\mathbb{P}$ and in $\overline{\mathbb{P}}$. We detail hereafter how we obtain such boxes in $\mathcal{C}_\mathcal{S}(t)$ for a given capture time t. Following common practice, we represent a bounding box coordinates by a quadruplet containing the two coordinates of the center of the box, its width and its height.

4.3.2 Surrogate ground truths inside the common region

This part is straightforward. First, we use $\mathcal{T}$ to detect players in $\mathcal{C}_\mathcal{T}(t)$ and retrieve the coordinates of bounding boxes of $\mathcal{C}_\mathcal{T}(t)$. Then, we project them into $\mathcal{C}_\mathcal{S}(t)$ using the calibration of the previous section. By doing so, we obtain the surrogate ground-truth bounding boxes of $\mathcal{C}_\mathcal{S}(t)$ that are located in $\mathbb{P}$, as shown in Figure 4.3. The remaining part of $\mathbb{P}$ constitutes detection-free areas.

4.3.3 Surrogate ground truths outside the common region

Dealing with the pixels outside the common part is difficult as $\mathcal{C}_\mathcal{T}(t)$ cannot feed the region $\overline{\mathbb{P}}$ with annotated bounding boxes. Training $\mathcal{S}$ solely with the boxes provided in $\mathbb{P}$ for each $\mathcal{C}_\mathcal{S}(t)$ leads the network to focus only on $\mathbb{P}$ and to overlook $\overline{\mathbb{P}}$ for each frame. Eventually, the network is not able to detect anything in $\overline{\mathbb{P}}$.

To circumvent this problem, our idea is the following. First, we use a custom data augmentation process to create artificial players with known bounding boxes in $\overline{\mathbb{P}}$. This provides us some surrogate ground-truth locations of some artificial true positive players that $\mathcal{S}$ will have to detect. This is not sufficient as we still need surrogate ground-truth information in areas where we did not create any player. For that purpose, we use a motion detection algorithm to identify areas of $\overline{\mathbb{P}}$ that are guaranteed player-free. This provides us hopefully true negative areas, in which $\mathcal{S}$ will be penalized when predicting player bounding boxes. In the remaining areas of $\overline{\mathbb{P}}$, we have no useful information, hence $\mathcal{S}$ will not be penalized. These two steps are described in detail in the two next sections.

4.3.3.1 Custom data augmentation

In order to introduce artificial true positive players with known bounding boxes in $\overline{\mathbb{P}}$, we design the following automatic data augmentation process. Given a frame $\mathcal{C}_\mathcal{S}(t)$, we start by randomly extracting image crops delimited either by one isolated or by several adjacent bounding boxes previously obtained in $\mathbb{P}$ (Figure 4.4). Then, for each crop, we randomly select a pixel in $\overline{\mathbb{P}}$, which will serve as an anchor point where the crop will be pasted after being rescaled and rotated appropriately. In our use case, the anchors are selected in the subset of $\overline{\mathbb{P}}$ corresponding to the soccer field.

We perform a rescaling and a rotation of each crop to produce an insertion that looks as realistic as possible by taking into account the inherent distortions of $\mathcal{C}_\mathcal{S}$ (Figure 4.4). Let (r,θ) denote the initial polar coordinates (with origin located at the center of $\mathcal{C}_\mathcal{S}(t)$) of the center of the crop and (r',θ') those of its selected anchor point. We rescale the crop by a factor $\alpha e^{\beta(r'-r)}+\gamma$ with $\alpha=0.5$, $\beta=-0.004$, $\gamma=0.5$ and rotate it by the angle difference $\theta'-\theta$. Finally, we paste the transformed crop on $\mathcal{C}_\mathcal{S}(t)$ itself with OpenCV's seamless blending function, such that its center is located at the selected anchor point (Figure 4.4).

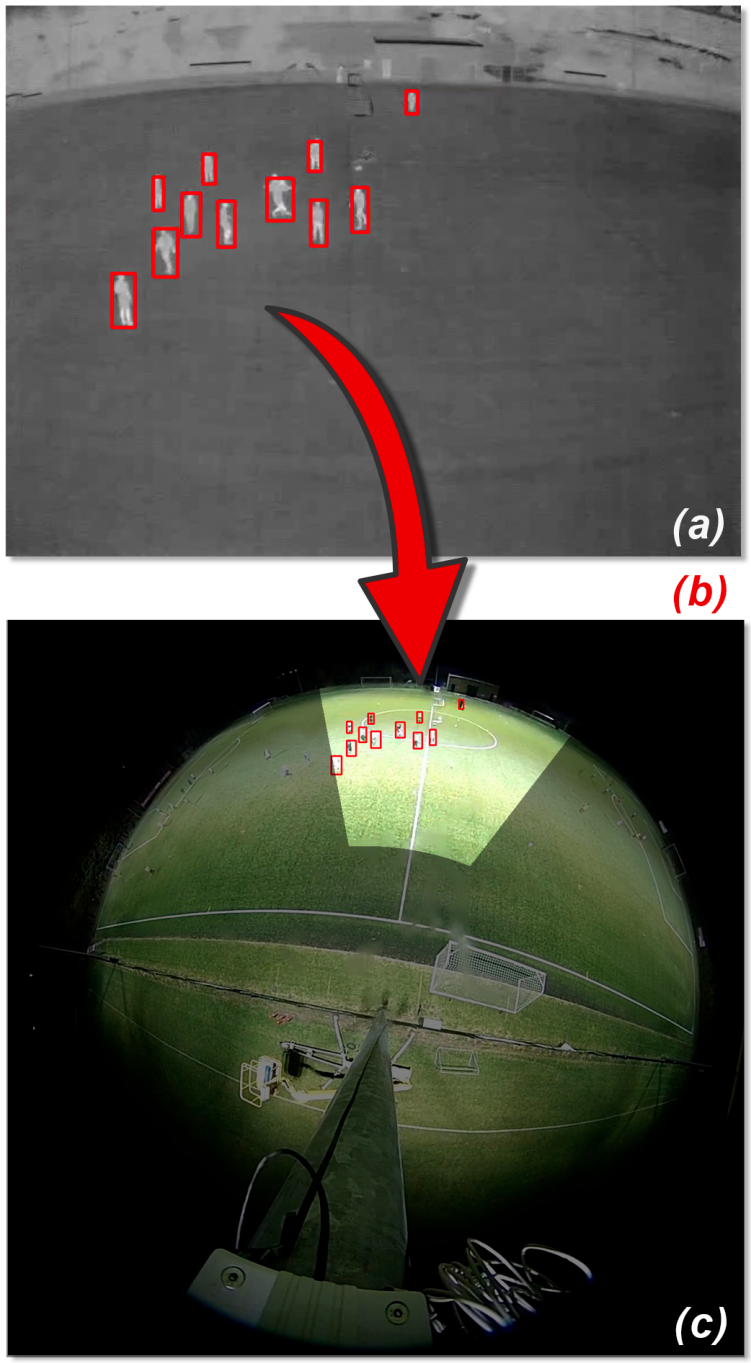


Figure 4.3: Bounding boxes transfer. The bounding boxes given by $\mathcal{T}$ on $\mathcal{C}_{\mathcal{T}}(t)$ (a) are projected (b) into $\mathcal{C}_{\mathcal{S}}(t)$ to provide us surrogate ground-truth bounding boxes in $\mathbb{P}$ (c).

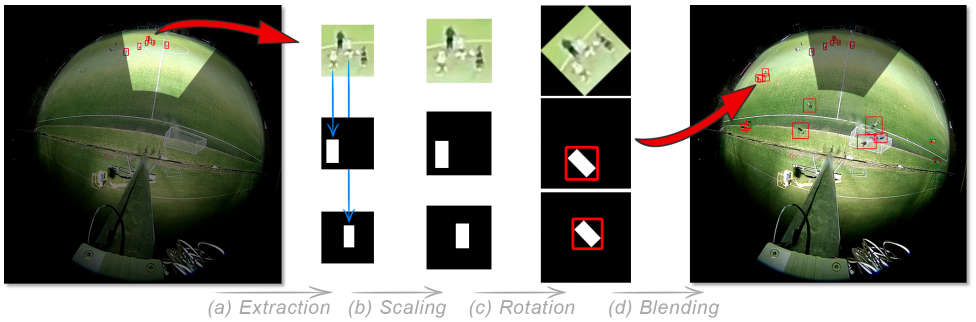


Figure 4.4: Our custom data augmentation pipeline designed to construct surrogate ground-truth bounding boxes in the region $\overline{\mathbb{P}}$ filmed by $\mathcal{C}_{\mathcal{S}}$ only. First, crops containing players are extracted (a) from the area filmed by both cameras $\mathbb{P}$, in which we know their location. Then, each crop and its associated bounding boxes are scaled (b) and rotated (c) to be appropriately pasted in $\overline{\mathbb{P}}$. A seamless blending is applied during the collage to increase the realistic aspect of the augmented image. As a result, we create artificial players with known bounding boxes in $\overline{\mathbb{P}}$.

In order to obtain the boxes associated with these artificial players, we perform the same transformation on each bounding box included in the initial crop. Eventually, for each transformed box, we consider as surrogate ground-truth bounding box the smallest unrotated (regular) rectangular box that encloses it (Figure 4.4).

In our fisheye setup, the data augmentation process allows to create artificial players with known bounding boxes in $\overline{\mathbb{P}}$ (Figure 4.4). However, this does not suffice to train $\mathcal{S}$ efficiently, as real players without known boxes may still be present in $\overline{\mathbb{P}}$. In a standard training process, $\mathcal{S}$ would thus be forced to detect the artificial players and would be penalized for detecting the remaining real ones. To bypass this undesirable effect, we remove the penalty suffered by $\mathcal{S}$ for detections containing enough motion. Hence, we leverage a background subtraction algorithm, as introduced in Chapter 2, to determine where this should be applied. By doing so, we also obtain areas where there is assuredly no player, *i.e.* where detections should not be made.

4.3.3.2 Leveraging background subtraction

As we handle a video feed from a fixed camera, we use ViBe [13] to obtain, for each frame $\mathcal{C}_{\mathcal{S}}(t)$, the set of pixels that are in motion, noted $\mathrm{B}(t)$, and those that are not, noted $\overline{\mathrm{B}(t)}$ (Figure 4.5). ViBe is very sensitive to motion, which implies that, in our fisheye setup, $\mathrm{B}(t)$ almost surely contains all the players, as well as pixels corresponding to the balls, player shadows, and some noise. As $\mathrm{B}(t)$ may be tight around the players, we apply a morphological dilation by a 11×11 square kernel on $\mathrm{B}(t)$ to ensure that it includes the bounding boxes that would surround the players if they were available (Figure 4.5). By doing so, we obtain an enlarged mask $\mathbb{B}(t)$, such that we can penalize $\mathcal{S}$ when it detects

players in $\overline{\mathbb{B}(t)}$, *i.e.* outside the enlarged mask. However, $\mathbb{B}(t)$ remains an area of uncertainty, where we do not penalize $\mathcal{S}$. Technically, this means that we zero out the loss in this area during training, as detailed hereafter.

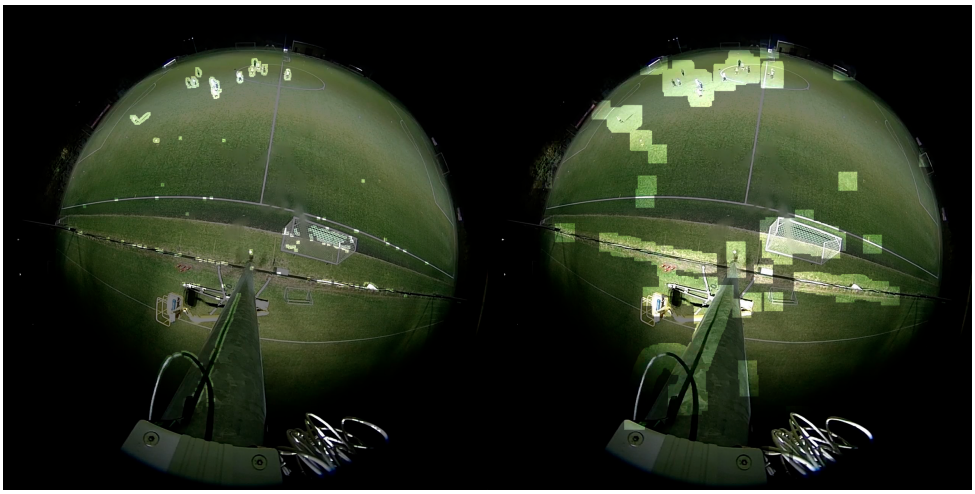


Figure 4.5: Motion masks. Initial motion detection mask $\mathrm{B}(t)$ overlayed on its corresponding frame (left), and enlarged motion detection mask $\mathbb{B}(t)$ (right).

4.3.4 Training the student

We use the YOLOv3 network [156] trained to detect humans on thermal images in [190] as teacher network $\mathcal{T}$. We use YOLOv3-tiny [156] as student network $\mathcal{S}$, adapted for a single class problem and with four times less channels for each convolutional layer to speed up its inference. Hence, $\mathcal{S}$ outputs a list of 5-dimensional vectors. Each of them encapsulates information on a predicted bounding box: the four coordinates (x, y, w, h) defining the box, and a player score p representing its confidence for a player to actually belong to the box.

The loss of YOLOv3-tiny, hence $\mathcal{S}$, penalizes these vectors in the following way (see Figure 4.6). For a predicted box close to a surrogate ground-truth box (either in $\mathbb{P}$ or in $\overline{\mathbb{P}}$), the mean square error loss between the coordinates of the boxes is computed, as well as the binary cross-entropy loss of p. This encourages the network to predict a high confidence score (closer to 1) and to find the right dimensions of the box. For a box far from a surrogate ground-truth box, only the binary cross-entropy loss of $1 - p$ is computed, to discourage the network from predicting a player in that box (p closer to 0). In our case, we must take into account the uncertainty about the boxes in $\mathbb{B}(t)$ in the region $\overline{\mathbb{P}}$, as they may correspond to unannotated real players. Therefore, for a box far

from a surrogate ground-truth box (including those created by the data augmentation), we zero out its loss if the center of the box is in $\overline{\mathbb{P}}$ and is in motion (belongs to $\mathbb{B}(t)$). If the center of the box belongs to $\overline{\mathbb{B}(t)}$, we are practically sure that there is no player in the box, and we thus leave the loss as is to penalize that detection. There is not particular restriction about the loss in $\mathbb{P}$. These different zones are illustrated in Figure 4.6.

4.3.5 Inference

When used for inference, we first apply a non-maximum suppression technique to remove duplicate bounding boxes with at least 50% of overlap and then verify that the bounding boxes predicted by $\mathcal{S}$ contain enough motion. Indeed, the predicted boxes whose center is not in motion, *i.e.* outside $\mathbb{B}(t)$, are not likely to contain a player. Therefore they are removed from the final output of $\mathcal{S}$. This allows us to remove potential false positive instances of players and improves the overall performance of the method.

4.4 Experiments

4.4.1 Experimental setup

The experimental setup is the following. We perform the distillation of the teacher network $\mathcal{T}$ into the student network $\mathcal{S}$ in an online manner as in Chapter 3. The reason for using that process is threefold. First, this allows $\mathcal{S}$ to continuously adapt to the latest weather and lighting conditions. Second, in a real-life deployment of the system, the online distillation will indeed be performed continuously. Hence, in order to have an understanding of how $\mathcal{S}$ behaves as it trains and detects people in real time, it is worth testing $\mathcal{S}$ under similar conditions. Third, training $\mathcal{S}$ adaptively allows us to study the evolution of the performance of the network as it learns through time. As we have only one video sequence with both the thermal and the fisheye recordings, this also enables us to evaluate $\mathcal{S}$ multiple times rather than measuring its performance only once, on a unique (and maybe abnormally hard or easy) small set of frames.

In the online distillation process, all the frames of the fisheye camera $\mathcal{C}_\mathcal{S}$ are treated by $\mathcal{S}$, which runs in real time. Meanwhile, some frames of the video feed of the thermal camera $\mathcal{C}_\mathcal{T}$ are input to $\mathcal{T}$, which provides boxes converted into surrogate ground-truth bounding boxes in the area $\mathbb{P}$ of the frame captured by $\mathcal{C}_\mathcal{S}$. These boxes are accumulated in an online dataset with 5-minutes memory, and the dataset is used to train a copy of $\mathcal{S}$ in a separate thread. The training is performed both on the $\mathbb{P}$ and $\overline{\mathbb{P}}$ regions of the $\mathcal{C}_\mathcal{S}(t)$ frames as described in the previous section, using our data augmentation and motion detection processes for region $\overline{\mathbb{P}}$. When this copy of $\mathcal{S}$ has trained during one epoch on the online dataset, its weights are updated and transferred into the initial network $\mathcal{S}$

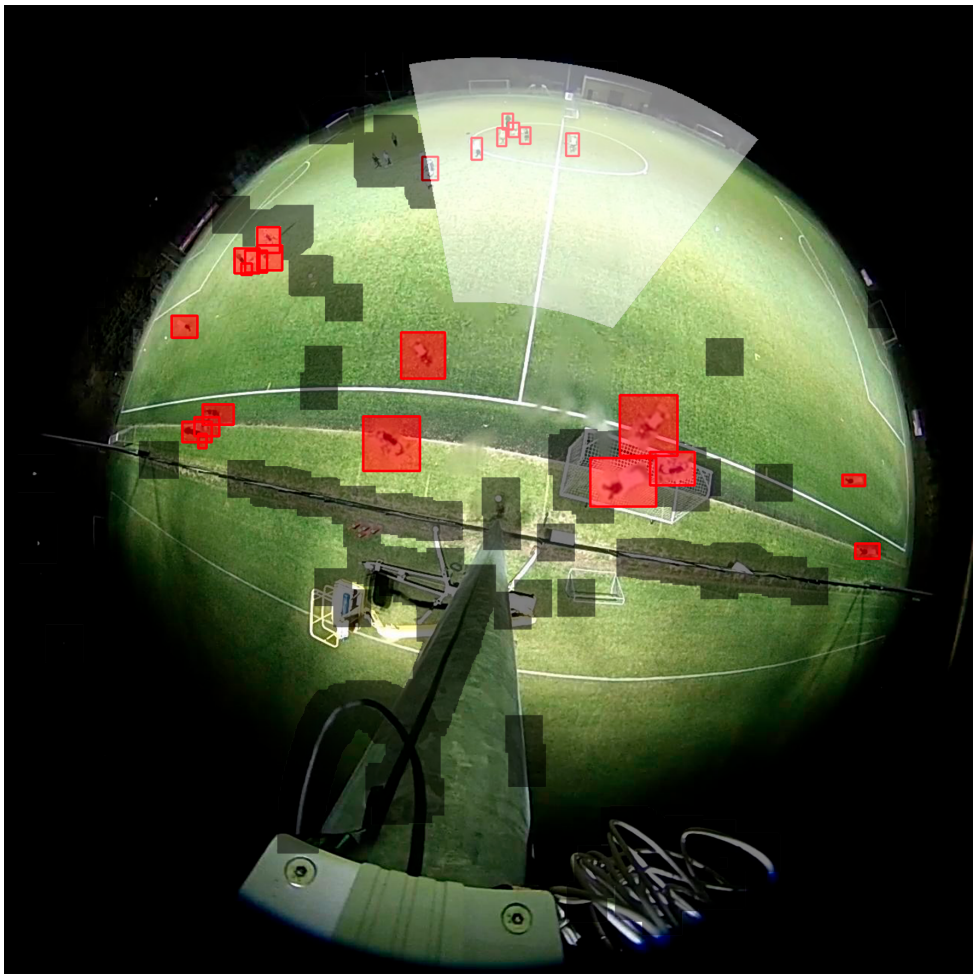


Figure 4.6: Combination of our data augmentation and motion detection algorithms, showing how the loss is applied to penalize the predictions of $\mathcal{S}$ in $\overline{\mathbb{P}}$ (outside the common area depicted in white). $\mathcal{S}$ must detect the players artificially created (red rectangles). Also, predicted boxes whose center falls within the enlarged motion mask $\mathbb{B}(t)$ (the black zones) do not generate any loss, since this area includes the players of $\overline{\mathbb{P}}$ not erased by the data augmentation, for which we have no ground-truth boxes. Finally, $\mathcal{S}$ must not predict any box in the rest of the image in $\overline{\mathbb{P}}$. Let us recall that the loss is applied everywhere in $\mathbb{P}$, as we have the ground truth from $\mathcal{T}$ in that area.

that performs the detection on all the frames. Consequently, the weights of this network evolves through time to continuously adapt to the latest video conditions.

4.4.2 Quantitative evaluation

To assess the performance of the student network $\mathcal{S}$ over the course of the video, we manually annotated the ground-truth bounding boxes for all the players of one frame every 10 seconds of the fisheye video. We compute the performance of $\mathcal{S}$ on a set of frames with the Average Precision (AP) metric particularized for one class. Following practice for the Pascal VOC dataset [55], each bounding box predicted by $\mathcal{S}$ is matched with the ground-truth box with which it has the largest intersection over union (IoU). We consider predicted boxes with an IoU larger than some threshold t_IoU as true positives, the others as false positives, and the ground-truth boxes left unmatched are false negatives. If several true positives are associated with the same ground-truth box, only one of them is kept as a true positive, while the others are rather considered as false positives. We note the number of true positives (respectively false positives, false negatives) TP (respectively FP, FN). Then, we compute the precision and recall as

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \quad \text{and} \quad \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{4.1}$$

We compute the points (R, P) for various thresholds on the confidence scores of the boxes to obtain the PR curve. Finally, we compute the area under the PR curve as suggested in [55] to obtain the AP for that set of frames. Despite its limitations [20], this kind of evaluation process has been widely adopted in the community.

In order to determine an appropriate value of t_IoU for evaluating the performance of $\mathcal{S}$, we examine the efficiency of $\mathcal{T}$ in predicting the boxes in $\mathbb{P}$. For that purpose, we compute the AP of $\mathcal{T}$ on the last 15 minutes of video, for several values of t_IoU ranging from 0 to 1, for the frames where ground-truth annotations are available. This allows us to determine how good $\mathcal{T}$ is at centering its bounding boxes on the players. The performance of $\mathcal{T}$ in $\mathbb{P}$ as a function of t_IoU is shown in Figure 4.7. We can see that $\mathcal{T}$ is not perfect in $\mathbb{P}$, which conditions the performances that can be expected from $\mathcal{S}$. To evaluate $\mathcal{S}$, we choose t_IoU= 0.25, as $\mathcal{T}$ displays reasonable performances in $\mathbb{P}$ with that threshold. Given the small size of the boxes, it also makes sense to examine the performance of $\mathcal{S}$ for a relatively low value of t_IoU. Let us recall that the boxes outputted by the network are independent of any particular choice of threshold. It serves only for quantitative evaluation purposes.

Following the methodology introduced in Chapter 3, we evaluate the performance of the student network $\mathcal{S}$ progressively. Every 10 seconds, $\mathcal{S}$ predicts the bounding boxes of the frames for which we have manual annotations within a running temporal window that covers the next 3 minutes of video. For this set of frames, we compute the AP. The

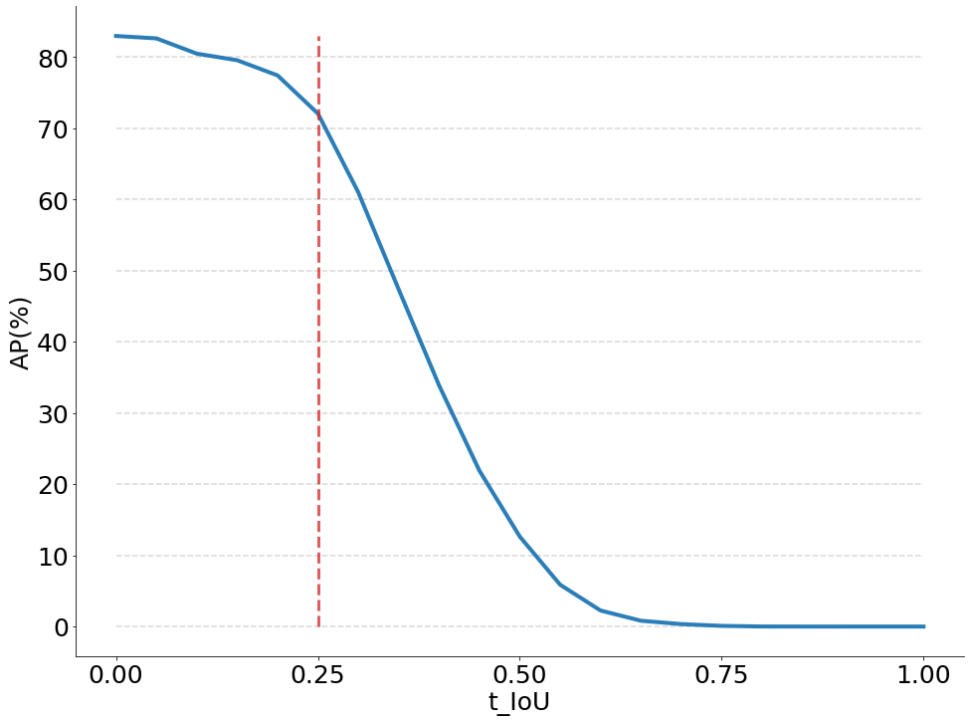


Figure 4.7: Choosing a t_IoU **for evaluation.** **Performances of $\mathcal{T}$ in $\mathbb{P}$** on the last 15 minutes of video as a function of t_IoU. This quantifies how accurately $\mathcal{T}$ centers its bounding boxes on the players. We can see that $\mathcal{T}$ is not perfect. We decide to evaluate the **performances of $\mathcal{S}$ for t_IoU=** 0.25, as we consider it as the largest t_IoU for which $\mathcal{T}$ still displays satisfying performances (AP $>$ 70%).

evolution on the AP through time with t_IoU= 0.25 is represented in Figure 4.8. We see that the performance tends to increase, indicating that $\mathcal{S}$ learns to better detect players over time. Figure 4.8 also reveals that there is still room for improvement in the present challenge.

We further examine the effectiveness of our data augmentation and motion detection processes to train $\mathcal{S}$ for detecting players outside $\mathbb{P}$. For that purpose, we perform a region-specific analysis by computing the temporal evaluation of the AP within $\mathbb{P}$ and/or $\overline{\mathbb{P}}$. The performance curves are displayed in Figure 4.8. We note that $\mathcal{S}$ learns efficiently to detect players in $\overline{\mathbb{P}}$, as the performances for $\mathbb{P}$ and $\overline{\mathbb{P}}$ are close to each other and follow the same trend. Also, further experiments reveal that the post-processing with the motion mask $\mathbb{B}(t)$ is particularly helpful to increase the performance in $\overline{\mathbb{P}}$. In that area, the AP decreases by 5 to 20% without post-processing, while the drop is below 3% in $\mathbb{P}$.

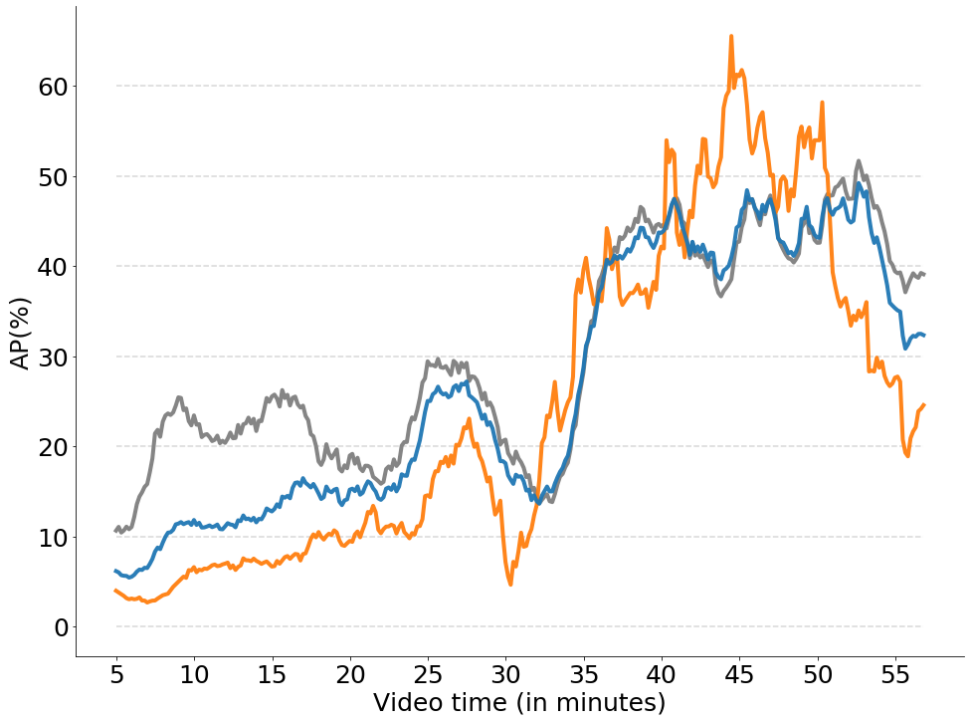


Figure 4.8: Evolution of the performances of the student network $\mathcal{S}$ through the video in $\mathbb{P}$, $\overline{\mathbb{P}}$, and in the **whole frames**. We can see that the network improves over time and that it manages to perform well both in $\mathbb{P}$ and in $\overline{\mathbb{P}}$.

Finally, as a potential application of this system is to monitor the use of the soccer field, we examine the results obtained for the task of people counting. The predicted number of people on the field corresponds to the number of bounding boxes predicted by $\mathcal{S}$ (thus on the fisheye images) after post-processing. We average the counting using a 1-minute sliding window. The results are displayed in Figure 4.9. We note that our method gives a globally reliable estimate of the number of people present on the field. Quantitatively, during the last 15 minutes of video, the root mean square error (RMSE) between the predictions and the ground truth is as low as 3.4 players. Again, we can see that the performance tends to increase over time since the estimate is more accurate at the end of the video, indicating that $\mathcal{S}$ learns to better detect players over time. Also, we can see in Figure 4.9 that the standard deviation of the number of detected players computed for each sliding window decreases over time, which indicates that the network becomes more consistent as it trains. Even though $\mathcal{S}$ tends to slightly overestimate the actual number of players, we can see that it manages to provides a good overview of the use of the field.

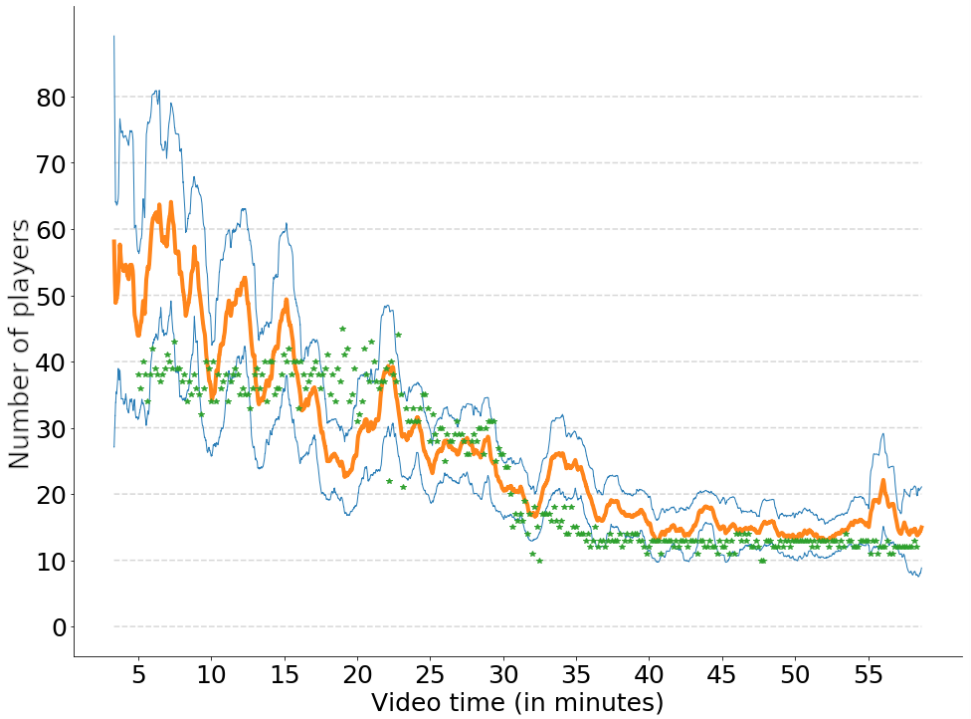


Figure 4.9: Counting performances. **Results on the player counting task** averaged over a 1-minute window, and associated **standard deviation**. During the last 15 minutes, we have a RMSE with the **ground truth** of 3.4 players, which is reasonable and shows that our method provides a reliable estimate of the occupancy of the soccer field.

4.4.3 Qualitative evaluation

To further assess the usefulness of our data augmentation and motion detection processes, we perform ablation studies on the components of our method. We investigate the combination of either enabling or disabling the data augmentation, with either zeroing out the loss in the motion mask $\mathbb{B}(t)$, or nowhere in $\overline{\mathbb{P}}$, or everywhere in $\overline{\mathbb{P}}$. The effects observed for these setups are reported in Table 4.1. In our experiments, we observe that the combination of the data augmentation and of zeroing out the loss in $\mathbb{B}(t)$, as detailed in Section 4.3, leads to the best student network $\mathcal{S}$ at inference time. Activating the loss everywhere in $\overline{\mathbb{P}}$ at training time forces $\mathcal{S}$ to detect only the artificial players in $\overline{\mathbb{P}}$ and to avoid detecting the actual players of $\overline{\mathbb{P}}$ that have not been erased by the data augmentation. This may confuse $\mathcal{S}$, leading to a decrease in its ability to detect players in $\overline{\mathbb{P}}$ at inference time. We notice that canceling the loss everywhere in $\overline{\mathbb{P}}$ leads to thousands of predicted bounding boxes in $\overline{\mathbb{P}}$ at inference time. This makes sense since the network is

not forced to detect or not players in $\overline{\mathbb{P}}$ in this case. Most of these predictions are false positives, and the system is useless in practice. As indicated in Table 4.1, we also note that removing the data augmentation always leads to mediocre networks, for similar reasons as those already explained. In particular, activating the loss everywhere in $\overline{\mathbb{P}}$ makes $\mathcal{S}$ unable to detect any single player in $\overline{\mathbb{P}}$. This results from the absence of ground-truth true positives (both artificial and real ones) in $\overline{\mathbb{P}}$.

In $\overline{\mathbb{P}}$	With data augmentation	Without data augmentation
Cancel loss in the motion mask $\mathbb{B}(t)$	**Our full method**. Most players in $\overline{\mathbb{P}}$ are correctly detected with few false positives.	Few players detected in $\overline{\mathbb{P}}$, unusable in practice
Activate loss everywhere in $\overline{\mathbb{P}}$	Able to detect some players in $\overline{\mathbb{P}}$, but not as good as our full method	unable to make any detection in $\overline{\mathbb{P}}$, no true positives
Cancel loss everywhere in $\overline{\mathbb{P}}$	Thousands of detections in $\overline{\mathbb{P}}$, mostly false positives	Thousands of detections in $\overline{\mathbb{P}}$, mostly false positives

Table 4.1: Ablation results in $\overline{\mathbb{P}}$. The combination of the data augmentation and the motion detection algorithm gives the best trade-off between true and false positive detections.

Finally, examples of detections provided by $\mathcal{S}$ are given in Figure 4.10. We can see that players located in $\overline{\mathbb{P}}$ are detected as efficiently as those located in $\mathbb{P}$. This was made possible thanks to our data augmentation and motion detection algorithms in the distillation approach.

4.5 Conclusion

In this chapter, we have proposed a novel system for monitoring the field occupancy in low-budget soccer stadiums. Our system uses a single wide-angle fisheye camera assisted by a thermal camera to detect and count all the players on the field. We use a network trained in a student-teacher distillation approach. The student network is locally supervised by a teacher network that easily detects players on the thermal camera. These detections are then projected into the fisheye camera using camera registration and serve as surrogate ground truths. Since both cameras have different modalities and fields of view of the scene, the student cannot be fully supervised by the teacher. Therefore, we develop a custom data augmentation process, combined with motion information provided by a background subtraction algorithm, to introduce surrogate ground truths outside their common field of view. In our case, we perform the distillation in an online fashion, *i.e.* our student is continuously trained to adapt to the latest video conditions, while performing the player detection in real-time. We show that our system is able to

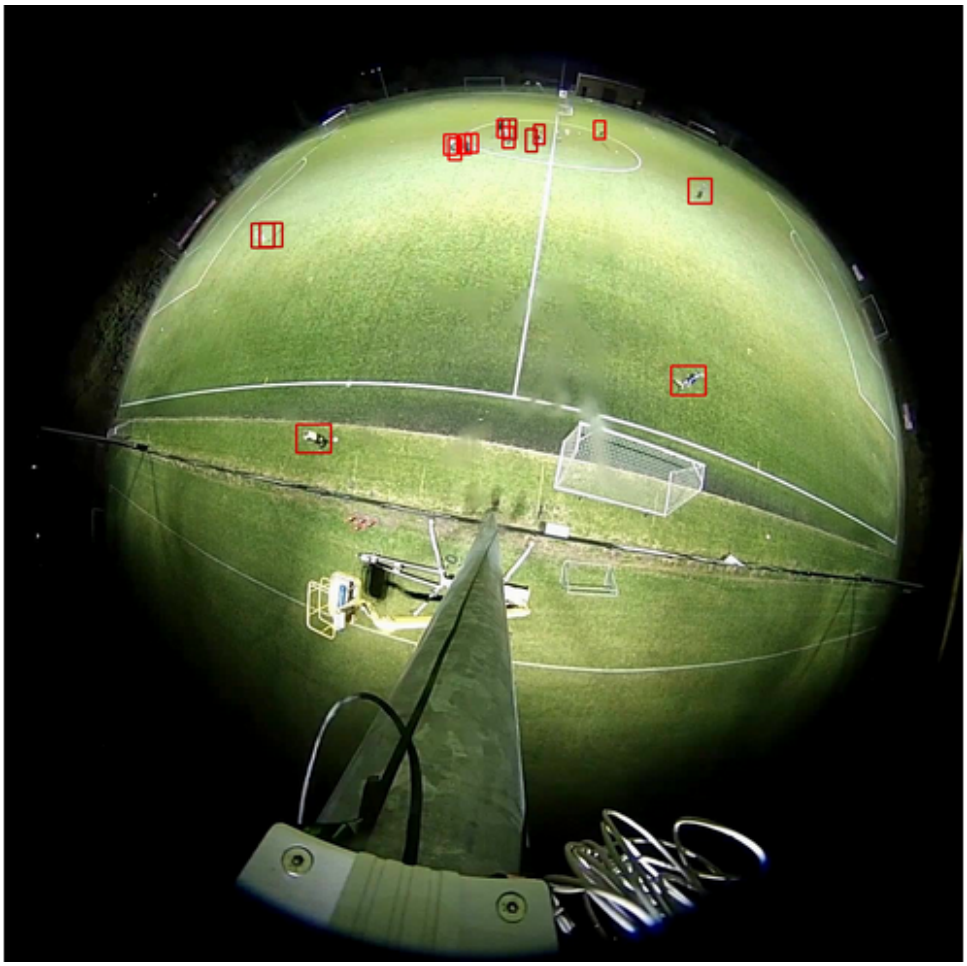


Figure 4.10: Detections on a test frame. We can note that players are accurately detected, even though there are a few superfluous predicted bounding boxes.

accurately detect players both inside and outside the common field of view, thanks to our custom supervision.

This chapter showed the generalization potential of our online distillation method to more challenging cases. We believe that online distillation can be very useful in many situations. In fact, we have seen in Chapter 3 that it can be used to perform a task in real time from non-real time algorithms and get close performances while adapting to the latest video conditions. Even though we presented it in the case of semantic segmentation, the same principles can be applied as is for many tasks, such as background subtraction, object detection or even panoptic segmentation. In this chapter, we showed another possible use of online distillation, when annotations are only available in another modality than the targeted one. Combining the findings of these two chapters, the possibilities on which online distillation can be applied are almost endless.

Part II
High-level semantics

CHAPTER 5

A bottom-up approach for high-level semantics

Contents

The chapters in Part I introduced three different types of low-level semantics, namely background subtraction, semantic segmentation and object detection. These low-level semantics provide a rather complete description of the video in terms of its environment, objects and motions. In this second part, we aim to provide high-level semantics about the interpretation of the soccer game. This second part focuses on the game phases in this chapter and action spotting of the game events in Chapter 6. Finally, in Chapter 7, we propose a novel dataset to boost the development of new automatic methods related to the interpretation of soccer games and presents new tasks for high-level semantics.

In this chapter, we describe a bottom-up approach to extract game phases in a soccer game based on low-level semantics. In particular, we extract three types of low-level semantics: a segmentation mask of the field, the field lines and the players. These segmentation masks are then processed to compute semantic features representative of the game phases and related to characteristics of the players or the camera view. For example, they correspond to how players are positioned in the image or the part of the field that is filmed. Finally, we show how these semantic features can be used to set up and train a semantic-based decision tree classifier for major game phases with a restricted amount of training data.

The main advantages of our approach are that it only requires the video feed of the main camera to extract the semantic features at the different levels, with no need for

camera calibration, field homography, player tracking, or ball position estimation. While game phases are only a subpart of automatic interpretation of sports games, it remains challenging and useful to the comprehension of soccer game. In fact, it can be at the basis of an attention mechanism for spotting interesting actions or to drag the attention of the viewer at key moments of the game. Our approach allows us to achieve promising results for game phases, which are expressed as a temporal segmentation between: goal or goal opportunity, attack, defense, and middle game.

This chapter is organized as follow. In Section 5.1, we introduce the problem of automatic interpretation of soccer games and provide some related work. Then, in Section 5.2, we describe our approach and explains the methods used to extract semantics at different levels. Finally, some experimental results are presented in Section 5.3 and Section 5.4 concludes this chapter.

PUBLICATION RELATED TO THIS CHAPTER

A. Cioppa, A. Deliège, and M. Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 1846–1855, Salt Lake City, Utah, USA, June 2018

Contribution. **(i)** We show a simple way to leverage low-level semantics to produce high-level semantics in the case of a soccer game. **(ii)** We propose the task of game phase segmentation. **(iii)** We propose an intuitive pipeline based on semantics for temporally segmenting game phases in a soccer game.

5.1 Introduction

In order to achieve the goal of automatic interpretation of soccer games, a first step can be to describe the content of the video with low-level semantics. A lot of research has been carried out for sports, like soccer, whose content is interpreted by starting with the extraction of particular features. In that spirit, several works have focused on the extraction of pedestrians using universal methods [42, 141, 195, 202], or players in the case of sports along with scene-specific tracking techniques [59, 67, 88, 132, 206]. Other works aim at extracting the position of the ball in various types of sports [88, 118, 146, 169] or compute a homography of the field [56, 75, 85, 137].

The analysis of game phases has also been addressed in projects such as Aspogamo [86], which is based on ball and players tracking methods using a field reconstruction model. Several works action spotting have also been conducted [50, 51, 53, 70, 105, 152, 214]. Finally, audio analysis of the excitement of commentators and detection of specific keywords in their speech has been investigated to detect important actions in [205].

Recently, with the emergence of deep learning, supervised classification has become more accurate and robust and can be used in the sports domain to capture information on the content of the image. Universal networks appeared with the spread of extensive annotated datasets such as ImageNet [49] or MS COCO [123]. Semantic segmentation networks such as PSPNet [221], Mask R-CNN [78], or PointRend [113] allow to segment any image into more than a hundred different classes. We have also shown that semantic segmentation can be used to improve background subtraction in Chapter 2. As we have seen in Chapter 3, universal networks are robust in many situations, but scene-specific networks are much better at segmenting the players on a soccer field.

This chapter presents a bottom-up approach based on semantics to temporally segment the game phases from the video stream of the main camera without the need for camera calibration, field homography, ball position estimation, or player tracking. First, we extract low-level semantics that describe the content of the video: the field, the players, and the lines. These low-level semantics are then used to extract semantic features that are representative of the game phases. We focus on semantic features that are also used by humans to understand the different soccer game phases. For that purpose, we analyzed several soccer games in order to grasp what makes us comprehend the different game phases. In particular, we noticed that characteristics of the players and the camera view are important features to take into account. For example, the part of the field that is shown or the way the players move. Based on these semantic features, we propose a simple semantic-based decision tree classifier for the major types of game phases: goal or goal opportunity, attack, middle game, and defense. An overview of our bottom-up approach is given in Figure 5.1.

5.2 A bottom-up approach for game phases segmentation

We aim at temporally segmenting the different game phases in a soccer video sequence from the main camera. As there exist no dataset for this high-level task (to train or evaluate a deep learning model), we choose to develop a hand-crafted approach based on low-level semantics and domaine knowledge about soccer. To clearly define our approach, we need to address the following three questions:

UNDERLYING QUESTIONS

Question 1. Which game phases do we want to segment?

Question 2. Which semantic features are relevant for identifying these game phases?

Question 3. Which approach should we use to segment these game phases?

With respect to Question 1, we choose to restrict ourselves to the segmentation of the four major game phases in a soccer game: attack, defense, middle game and goal or goal

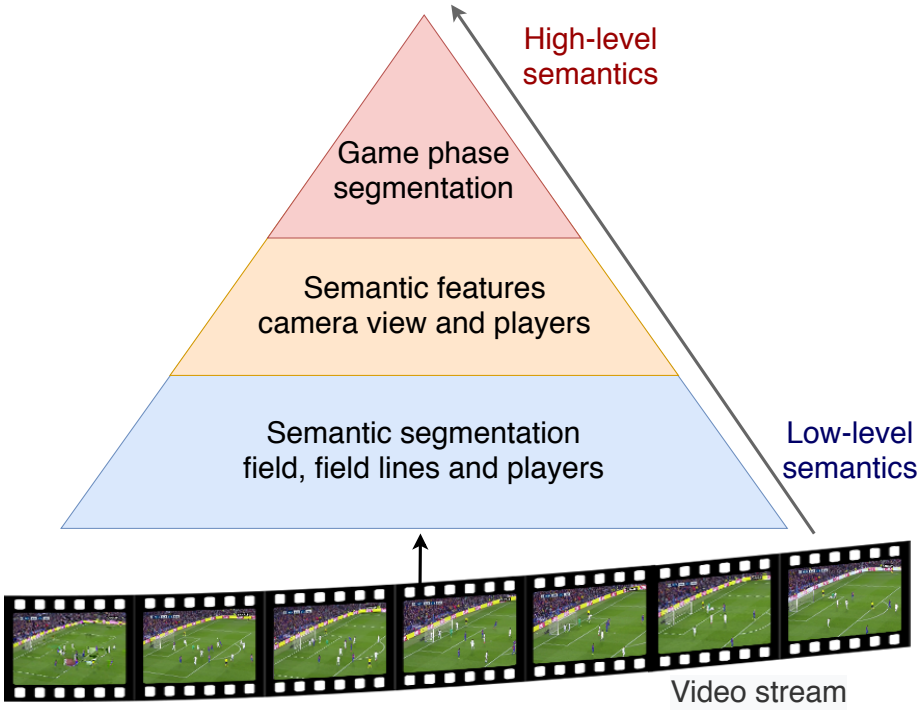


Figure 5.1: Overview of our scene-specific bottom-up approach to extract semantic features from the video stream of the main camera in a soccer game. Low-level semantics such as the field, field lines and the players are used to extract semantic features representative of the game phases. These are then used to temporally segment the different game phases: goal or goal opportunity, attack, middle game, and defense.

opportunities. These game phases are relevant for the interpretation of the soccer game as they provide information about the intention of the team possessing the ball. These game phases have unclear temporal boundaries, unlike game events such as corner, free-kick, cards, or substitutions which have a more precise definition in the soccer rule book and are covered in Chapter 6. Therefore, we provide our own definition of these game phases hereafter:

DEFINITION OF THE GAME PHASES

1. *Attack*: when the team possessing the ball is approaching the great rectangle or the goal of the other team with the intention of scoring.
2. *Defense*: when the team possessing the ball is close to its own great rectangle, with no intention of going towards the other team's side of the field.
3. *Middle* game: when the teams possessing the ball is close to the center of the field and the players simply pass the ball from one to another with no real intention of going towards the opposing team's goal.
4. *Goal* or *goal opportunity*: when there is an interesting action that could lead or has led to a goal, *e.g.*, a shot on target or a dangerous free-kick.

Then, to address Question 2, we identify which semantic features are representative of the aforementioned game phases. To do so, we manually analyzed several soccer games and determined what visually discriminates the different game phases. From our observation, we conclude that we can accurately separate the four major soccer game phases with the following types of semantic features:

SEMANTIC FEATURES REPRESENTATIVE OF THE GAME PHASES

1. *Camera view information*: indicating which part of the field is filmed. We choose a ternary indicator corresponding to which portion of the field is filmed (center, side or goal) and a binary indicator corresponding to the presence or absence of the main circle in the image.
2. *Player information*: A continuous group measure corresponding to how much the players are close relative to each other and a ternary motion indicator corresponding to the average motion in the horizontal direction of the players (towards the center of the field, towards the goal or static position).

Regarding Question 3, we choose a semantic-based decision tree to segment the different game phases based on the semantic features presented hereinbefore. This can be considered as a bottom-up approach as we first extract low-level semantics and move to higher semantic levels up to the game phases. The remaining of this section is organized as follows. First, we detail the extraction of low-level semantics in Section 5.2.1. Then, we show how we extract the semantic features representative of the game phases in Section 5.2.2. Finally, we explain the choices behind our semantic-based decision tree to temporally segment the game phases in Section 5.2.3.

5.2.1 Low-level semantics

We first explain how we extract the three types of low-level semantics from the video, namely *field*, *field line* and *player* segmentation masks. The segmentation mask of the field is obtained obtained using a dynamic thresholding method based on the hue histogram of the image, while line and player pixels are obtained using a deep learning semantic segmentation network using few annotations. These segmentation masks are then used to compute the semantic features representative of the game phases.

5.2.1.1 Semantic segmentation of the field

Segmenting the field is a valuable information since the portion of the field that is filmed is an indicator of where the game is taking place. This topic has already been investigated in many works. In [88], a static field model has been used to extract the field, where the mean color is computed given prior statistics calculated on a large dataset. The method presented in [169] is based on the peak of the histogram of the three RGB channels in order to compute some field parameters. An adaptive field model using the peak in the green chromaticity space has also been investigated in [150] for the particular case of robot soccer. Our segmentation method builds upon those methods and consists in computing a field model based on the color of the field, and then comparing each pixel to this model. It is important to note that we are interested in the field segmentation mask as if they were no players present on the field.

First, we change the color space of the original image from RGB to HSV which is more informative on the color of a pixel. To compute the field model, we hypothesize that the field is present in a large part of the image, which is a reasonable assumption for images provided by the main camera. Since the hue component of the field is homogenous within the image, we compute a hue histogram of the whole image and search for the highest peak in it. This peak should be located close to the mean value of the hue component of the field. We then threshold the image around this peak with a fixed width. This results in an intermediate segmentation mask containing all the pixels whose value matches that of the field model. Finally, several post-processing operations are applied on the intermediate segmentation map obtained at the previous step. First, morphological opening and closing operations with a 15×15 square structuring element are performed in order to remove small isolated objects located outside and inside the field, respectively. Then, the contours of all the objects within that map are computed. From these contours, we choose the one that encompasses the greatest area to be the field and compute an approximation of this contour using the method presented in [180], which is implemented in the OpenCV computer vision library [21]. This removes small bumps on the contour. We also compute the convex hull of this contour to remove small gaps on it. Finally all the pixels inside this contour are labeled as field pixels. The steps of this method are illustrated in Figure 5.2. Let us note that the histogram is recomputed for each frame in order to have a dynamical field model robust to illumination changes.

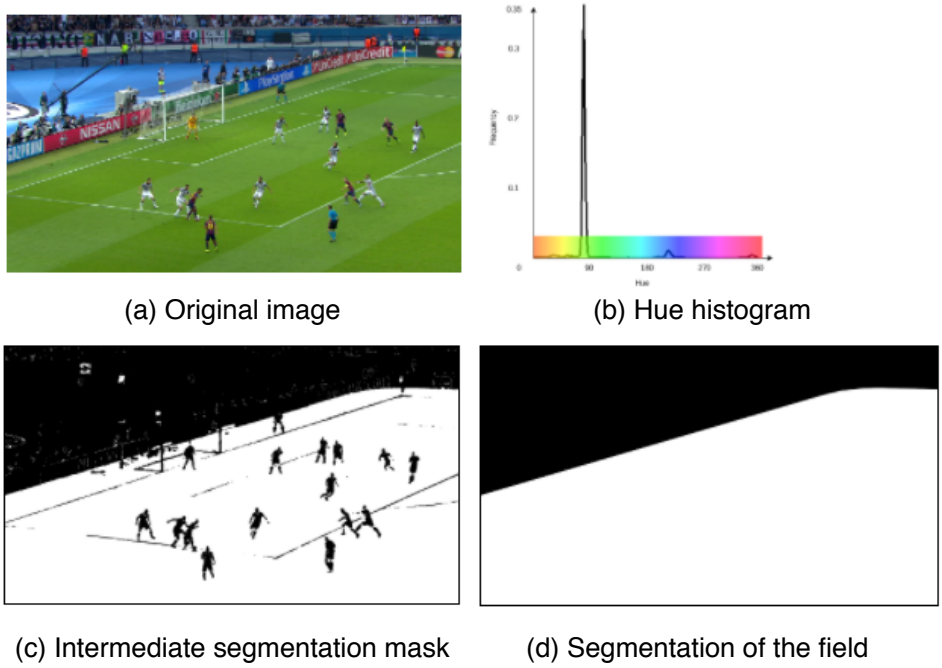


Figure 5.2: Steps of the field segmentation method. (a) Original image from which we segment the field. (b) Hue histogram of the image. The peak corresponds to the dominant color, which we assume to be the mean color of the field. (c) Intermediate segmentation mask obtained by thresholding the original image in its hue representation around the peak found in the histogram. (d) Final field segmentation mask of the field pixels obtained using post-processing operations on the intermediate segmentation mask. These consist in morphological opening and closing operations, contour and area selection, contour approximation, and convex hull.

It is important to note that the audience in the stadium sometimes represents half of the image, but its colors have a greater variance than those of the field. In practice, we noticed that the peak in the hue histogram still corresponds to the mean hue component of the field even when the field is shown in less than 20% of the global image, which is due to its small variance in the hue component.

5.2.1.2 Semantic segmentation of the field lines and the players

As stated in Chapter 1, scene-specific networks tend to perform better in the case of soccer, but they can require manual annotations which are laborious and time-consuming to obtain since each pixel has to be labeled individually. In Chapter 3, we designed an online distillation process to circumvent the annotation problem while making the network

adapt to the latest game conditions. But this method relies on the availability of a universal teacher network able to perform the segmentation. In the case of soccer field lines, their exist no such off-the-shelf network, meaning that we can not use online distillation and have to stick with manual annotation or use completely unsupervised algorithms.

Since annotating accurately each pixel belonging the lines would require too much effort, we decide not to implement a fine-grained semantic segmentation and explore another possibility in this chapter. We start by redefining our segmentation problem from pixels belonging exactly to a line to pixels that are in the neighborhood of a line. This approximate segmentation suffices to extract our semantic features presented in Section 5.2.2. This drastically speeds up the annotation process since it can now be done with rough blobs. For the semantic segmentation of the players, both online distillation and this approximate method can be used. We choose to go with the same method as the ones for the field lines. An example of annotation and the network's segmentation for the lines and players can be seen in Figure 5.3.

Then, we need to choose an the architecture for the segmentation network. We use the same one as the student network presented in Chapter 3, called TinyNet that was developed during this thesis. This network has the advantage of being lightweight and therefore real time. Furthermore, we showed that it can be trained efficiently with few annotations as it reaches high performances very quickly during the online distillation process, even if it is initialized from scratch. To do so, we only annotate 200 images of field lines and players, which can be done during a single day of work. For further details, a complete description of TinyNet can be found Section A.1.

5.2.2 Moving to higher levels of semantics

In the previous section, we designed methods to extract low-level semantics from soccer images with limited data. We now present the methods developed to extract semantic features representative of game phases. As a reminder, the choices of these semantic features is entirely motivated by the observation of how we, as humans, interpret the game. First, we choose to extract information about the camera view and the presence or absence of the main circle from the field and field line segmentation mask. Second, we extract a group measure of the players, the average position of the players, and the average direction of the players' motion from the players segmentation mask.

5.2.2.1 Camera view

The first semantic feature that we extract is a class corresponding to the part of the field that the camera is filming. We define three different classes of possible views which are displayed in Figure 5.4 along with their corresponding field segmentation mask:

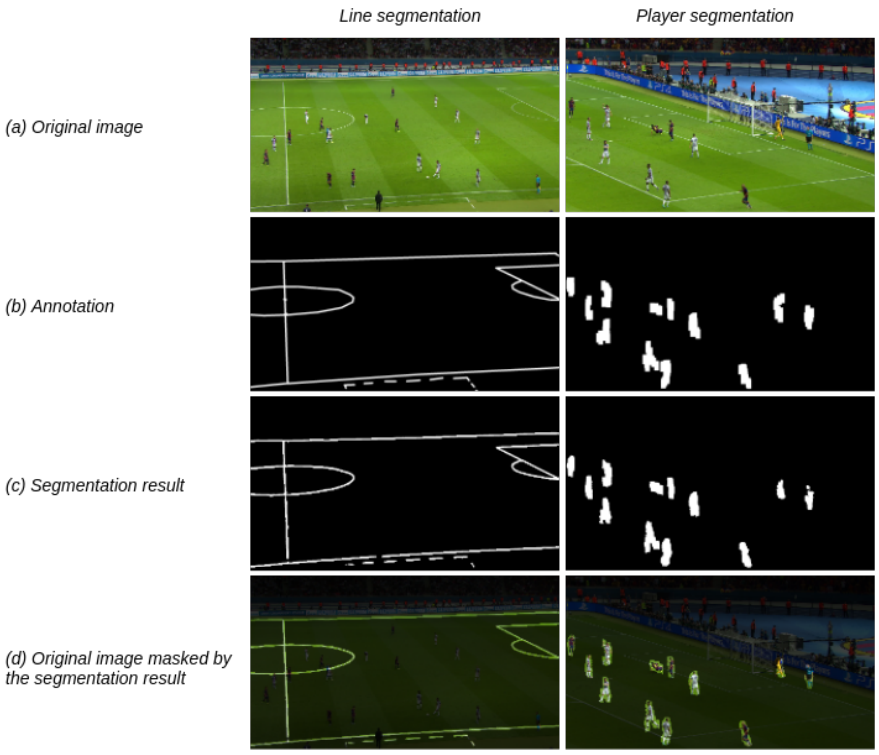
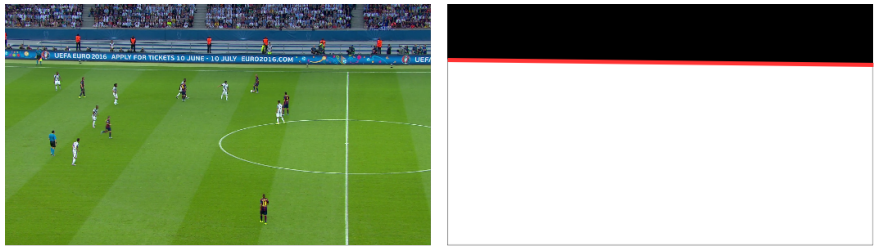


Figure 5.3: Semantic segmentation results. (a) Original image on which to segment the lines and players. (b) Annotations corresponding to pixels that are in the neighborhood of players or lines in order to increase the speed of the annotation process. (c) Segmentation results obtained with our scene-specific semantic segmentation network. (d) Original image overlaid by the segmentation result for visualization purpose.

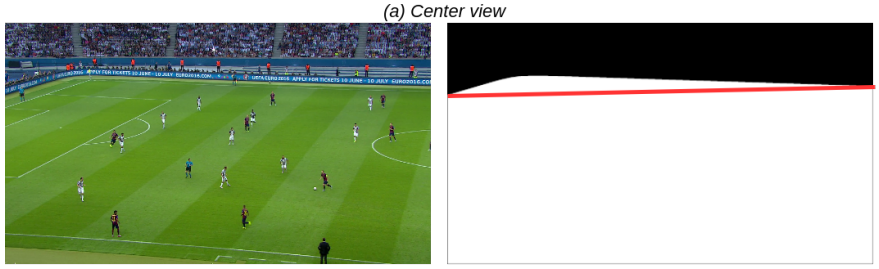
DEFINITION OF THE CAMERA VIEWS

- *Center view*: the camera is mainly centered on the field. It is characterized by a single separation in the field segmentation mask which has a small angle compared to the horizontal.
- *Side view*: the camera is clearly filming one side or the other of the field. It is characterized by the presence of a corner in the field segmentation map. Therefore, a first step to discriminate it is to detect the presence of such a corner.
- *Goal view*: the camera is zooming in on one of the two goals. It is characterized by a single separation in the field segmentation map which is diagonal.

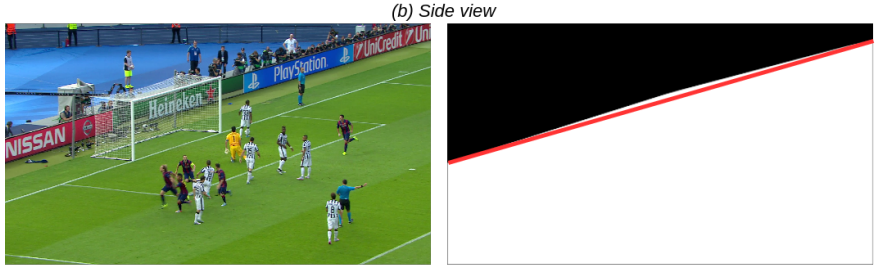
As described above, a first step is to compute the presence of a corner or the orientation of the separation. To do so, we start by computing what we call the *characteristic line* of the field segmentation mask which is defined as the line that joins the two uppermost intersections between the black and white separations at the edges of the mask. Examples of such characteristic lines are shown in red in Figure 5.4. To detect the presence of a corner in the field segmentation mask, we compare the ratio of segmented pixels on each part of the characteristic line. If the ratio is greater than some learned threshold, then we consider that there is a corner and the view is classified as a side view. If no corner is found in the previous step, then the tilt of the characteristic line is computed. If this tilt is greater than some other learned threshold, then it is a goal view; otherwise it is a center view.



(a) Center view



(b) Side view



(c) Goal view

Figure 5.4: Types of camera view. Examples of the three different types of camera views (left) and their corresponding field segmentation mask (right) with the characteristic line shown in red. (a) Center view with its characteristic horizontal line of separation. (b) Side view, with the presence of a corner in the segmentation mask. (c) Goal view, with its diagonal separation.

The second semantic feature that we extract is the main circle which is at the center of the field. This main circle is projected as an ellipse on the main camera and can be detected from the field line segmentation mask extracted in Section 5.2.1.2. Therefore, we choose to extract this ellipse using a RANSAC algorithm [60] which is a typical choice in many works for geometric fitting [94, 136].

The principle of RANSAC is to randomly select a subset of the data points and to fit a model on this subset. Then, all the data points are tested through a distance threshold function that determines whether a particular point agrees with the created model or not. The points that agree with the model are called inliers. Finally, a score function evaluates how well the model fits the original data. This procedure is repeated several times and the model that has the greatest score is selected and re-estimated using all the inliers of its model. In our case, the distance function is a threshold on the distance of a point to the ellipse, and the model is built using the robust ellipse fitting method of Fitzgibbon [61]. Finally, we check that this ellipse is in the range of plausible ellipses. Otherwise, we consider that the main circle is not shown in the image.

5.2.2.2 Player semantics

The last semantic features are extracted from the player segmentation mask computed in Section 5.2.1.2. The first one is the barycenter of the players which represents the mean position of the players on the image plane. It is straightforward to get such information since it simply consists in computing the barycenter of all segmented pixels. The fact that the segmentation mask represent pixel that are close to players is not a problem since these surrounding pixels are spread uniformly around the players and therefore do not affect the position of the barycenter.

Then, we compute some group measure that represents how much the players are close one to another. To estimate this group measure, we compute the mean squared distance of each point in the segmentation mask to the barycenter. The players are considered as grouped when this value is smaller than some learned threshold and as scattered otherwise.

For the last semantic feature, which is the average horizontal direction of the players on the field, we need to compute the horizontal difference between the barycenter of the players and a reference point on the field. Since we do not compute a homography or use calibrated cameras, we use one of the two following reference points, depending whether which one is visible in the image: the center of the main circle that we extracted or the end of the field segmentation map behind the goals. We can then simply look at the evolution of the distance of the barycenter compared to this reference point and check if it gets closer to one of the two goals or to the center of the field.

5.2.3 High-level semantics

From the semantic features extracted in the previous section, we design a semantic-based decision tree to temporally segment the game phases. We decide to avoid working with fully supervised machine learning techniques since annotated data of game phases are hard to collect. On top of that difficulty, it is often troublesome to have precise boundaries on the game phases since the transition is often blurry, such that the choice of a frame splitting two consecutive game phases is not unique. For this reason, we find it appropriate to design a decision tree with a fixed hand-made structure induced by the semantic features of the soccer game. In this way, we ensure that the tree structure does not overfit the limited amount of training data and that its interpretation is consistent with an intuitive human interpretation of the game. Besides, this leaves room for further extensions based on additional semantic features. After some tests and hours of watching soccer games, we built the final decision tree structure displayed in Figure 5.5. Given the restricted number of trainable thresholds to determine, they are learned by minimizing the segmentation error using a rapid grid search procedure.

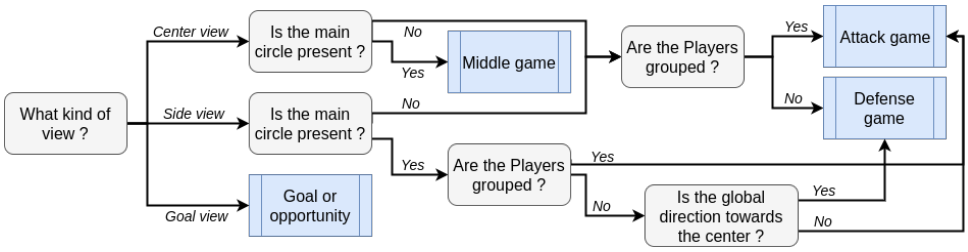


Figure 5.5: Structure of the semantic-based decision tree. The branches are based on the semantic features that we believe to be representative of game phases and that can be robustly obtained from low-level semantics such as the segmentation of the field, the players and the field lines.

5.3 Experiments

5.3.1 Evaluation methodology

We evaluate both the extraction of the low-level semantics and the game phases segmentation on the 2015 Champions League final soccer game. The data consists of a single full-HD video stream from the main camera. The deep learning models for semantic segmentation are trained on 180 images and tested on 20 using a classical training procedure. For the evaluation of the decision tree, we trained and determined its thresholds (5 of them) on one half time of the game, and evaluated the performances on the second half. Finally a temporal smoothing with a window of 10 frames is applied on the

semantic features and the game phases segmentation using either a majority vote when the indicator is discrete or a moving average when it is continuous.

5.3.2 Evaluation of semantic segmentation

We first evaluate the field segmentation masks by evaluating the proportion of the surface of the actual field that is covered by the segmentation, *i.e.* the recall:

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{5.1}$$

We also evaluate the ratio of the field that is uncovered, which is represented by the specificity:

$$\mathrm{S} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}, \tag{5.2}$$

and we define the accuracy as follows:

$$\mathrm{A} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{5.3}$$

where TP is the number of true positives, TN the number of true negatives, FP the number of false positives and FN the number of false negatives. The performances are evaluated on approximately 25 million pixels. As can be seen in Table 5.1, almost all the field is covered and the surroundings are rarely mistaken as field pixels as well. Also, a visual inspection of the results shows that our approach works well. Small errors occur only at the exact edge of the field which is not critical in our approach.

Second, we evaluate the performances of the semantic segmentation networks for the lines and players. Since we made a huge scale down compared to PSPNet, our network can be trained at a rate of 250 ms/image which corresponds to 45 seconds/epoch on our 180 image dataset. All these speed-ups allow an operator to fine-tune the network with a few images annotated during the warm up period in order to make the network more scene-specific and improve its performances. Also, we notice that impressive results can already be obtained with 180 images of a single game as training set with our network trained from scratch, which is a similar result to what was observed in Chapter 3.

We define our semantic segmentation task as labeling pixels are in the neighborhood of lines or players rather than belonging exactly to a line or a player. Thus, we need a performance indicator that evaluates the proportion of correctly classified line or player pixels among all the pixels classified as line or players as it is important not to have too many false positives outside the neighborhood polluting the segmentation map. This criterion is called the precision and is defined as

$$P = \frac{TP}{TP + FP}. \tag{5.4}$$

Table 5.2 shows some performance indicators for the lines and players segmentation which are evaluated on approximately 45 million pixels. As can be seen, the performances are satisfying even if we only provide few training images. This confirms that an operator can annotate a few frames before the game in order to make the network more scene-specific and improve its performances compared to universal networks. The training is also fast as it can be done at a rate of 45 seconds/epoch with impressive results achieved after only 75 epochs with a network trained from scratch on a single GPU. Furthermore, the modifications allow the network to segment full-HD images coming from the main camera at a framerate of 25 images per second on a single GPU, including transfer times.

Performance indicators	**Field pixels**
Recall (R)	0.997
Specificity (S)	0.990
Accuracy (A)	0.994
Computation time	30 ms/image

Table 5.1: Performance indicators for our field pixel extraction method using the hue histogram of the image. The results are evaluated on approximately 25 million pixels.

Performance indicators	**Player pixels**	**Line pixels**
Precision (P)	0.904	0.819
Accuracy (A)	0.989	0.987
Computation time	78 ms/image	

Table 5.2: Performance indicators for our scene-specific semantic segmentation network of the performance indicators for player and line pixels. The results are evaluated on approximately 45 million pixels.

5.3.3 Evaluation of the segmentation of game phases

In order to evaluate quantitatively the whole system, we annotate different clips of the video with their corresponding game phases. We train the thresholds of the decision tree based the first half and evaluate the performance on the second half. We reach an accuracy of 91.8%, which is an already impressive result for a method that is based on few training data.

Apart from the quantitative evaluation, it is also interesting to analyze the results qualitatively. In fact, the annotations do not always correspond to clear game phases

because sometimes it is difficult to assign a unique label for frames that separate two consecutive game phases. For example, the difference between an attack and a middle game can be tricky since an attack can start from the middle of the field. Thus, two annotators could assign a different label to the same game event. For that purpose, it is important to look at the output sequences and have a global qualitative estimation of the system as well. We notice satisfying results, which could be accurate enough for an attention based mechanism or an automatic alert system when an interesting game phase is taking place.

5.4 Conclusions

In this chapter, we have presented a bottom-up approach to extract semantic features from the video stream of the main camera in a soccer game. Our approach does not need camera calibration, field homography, player tracking, or ball position estimation, since these are often hard to get robustly. We first extract low-level semantics by segmenting three types of classes in the image: the field, which is found using a thresholding method based on the hue histogram, and the field lines and players which are extracted using our novel, tailored, scene-specific deep learning semantic segmentation network. Then, we extract semantic features based exclusively on important characteristics of the players and the camera view. From these features, we show that it is possible to learn the parameters of a simple semantic-based decision tree to segment the following major soccer game events: goal or goal opportunity, attack, middle game and defense, for which we got over 90% of accuracy.

The approach that we have presented in this chapter is far from being a solved subject of research. The list of semantic features and game phases that we presented is not exhaustive. With an increased amount of labeled data, it should be possible to improve both the extraction of semantic features and the decision mechanism for game phases segmentation. For instance, the structure of the decision tree could be learned. More sophisticated models such as deep learning temporal networks could also be used. In the next chapter, we focus on game events, also called actions, rather than game phases and propose a deep learning approach based on the natural context surrounding these actions rather than pre-computed low-level semantics.

CHAPTER 6

A context-aware loss function for action spotting

Contents

In the previous chapter, we used low-level semantics to extract the game phases in soccer videos, which correspond to high-level semantics about the game. In this chapter, we focus on game events, or actions, rather than game phases. These actions correspond to punctual events in the game such as goals, cards and substitutions. This task, called *action spotting*, was recently introduced by the SoccerNet dataset [72] and its objective is to detect actions in the TV broadcast. These actions are annotated as a single spot, which contrasts with game phases, which have a temporal duration.

In the following, we present a novel loss function that specifically deals with the temporal context around each action, rather than focusing on the single annotated frame to spot. We benchmark our loss on the large dataset of soccer videos SoccerNet, and achieve an improvement of 12.8% over the baseline and take the first place across all benchmarked methods. We also show the generalization capability of our loss for generic activity proposals and detection on ActivityNet, by spotting the beginning and the end of each activity. Finally, we provide a real-world use case for such high-level semantics by designing an automatic highlights generation method.

This chapter is organized as follows. Section 6.1 introduces the problem of action spotting and some related works are discussed in Section 6.2. Then, Section 6.3 details our loss and network for the action spotting task. Experimental results are provided in Section 6.4 along with an extended ablation study. Challenging cases for action spotting in soccer videos and an application of our method in automatic highlights generation is shown in Section 6.5. Finally, we conclude on this method in Section 6.6.

PUBLICATION RELATED TO THIS CHAPTER

A. Cioppa, A. Deliège, S. Giancola, B. Ghanem, M. Van Droogenbroeck, R. Gade, and T. Moeslund. A context-aware loss function for action spotting in soccer videos. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13123–13133, Seattle, Washington, USA, June 2020

Contributions. **(i)** We present a new loss function for temporal action segmentation and the task of action spotting, which is parameterized based on the context surrounding the actions. **(ii)** We show that our network architecture combined to our loss function improves the previous state-of-the-art performance for the action spotting task of SoccerNet by 12.8% **(iii)** We provide detailed insights into our action spotting performance, and showcase a possible application for the automatic generation of highlights.

6.1 Introduction

Aside from automotive, consumer, and robotics applications, sports is considered as one of the most valuable applications in computer vision [178], capping $91 billion of annual market revenue [107], with $28.7 billion from the European Soccer market alone [48]. Recent advances helped provide automated tools to understand and analyze broadcast games. For instance, current computer vision methods can localize the field and its lines [56, 85], detect players [35, 206], their motion [58, 132], their pose [24, 215], their team [97], track the ball position [168, 182] and the camera motion [127]. Understanding spatial frame-wise information is useful to enhance the visual experience of sports viewers [158] and to gather players statistics [183], but it misses high-level game understanding. For broadcast producers, it is of paramount importance to have a deeper understanding of the game actions. For instance, live broadcast production follows specific patterns when particular actions occur; sports live reporters comment on the game actions; and highlights producers generate short summaries by ranking the most representative actions within the game. In order to automate these production tasks, computer vision methods should understand the salient actions of a game and respond accordingly. While spatial information is widely studied and quite mature, localizing actions in time remains a challenging task for current video understanding algorithms.

In this chapter, we target the action spotting challenge, with a primary application on soccer videos. The task of action spotting has been defined as the temporal localization

of human-induced events annotated with a single timestamp [72]. Inherent difficulties arise from such annotations: their sparsity, the absence of start and end times of the actions, and their temporal discontinuities, *i.e.* the unsettling fact that adjacent frames may be annotated differently albeit being possibly highly similar. To overcome these issues, we propose a novel loss that leverages the temporal context information present around the actions, as depicted in Figure 6.1. To highlight its generality and versatility, we showcase how our loss can be used for the task of activity localization in ActivityNet [81], by spotting the beginning and end of each activity, using the network BMN introduced in [120] and simply substituting their loss with our enhanced context-aware spotting loss function.

6.2 Related work

6.2.1 Soccer video understanding

Computer vision tools are widely used in sports broadcast videos to provide soccer analytics [134, 183]. Current challenges lie in understanding high-level game information to identify salient game actions [37, 187], perform automatic game summarization [167, 171, 189] and report commentaries of live actions [211]. Early work uses camera shots to segment broadcasts [53], or analyze production patterns to identify salient moments of the game [157]. Further developments have used low-level semantic information in Bayesian frameworks [87, 181] to automatically detect salient game actions.

Machine learning-based methods have been proposed to aggregate temporally hand-crafted features [9] or deep frame features [100] into recurrent networks [152]. SoccerNet [72] provides an in-depth analysis of deep frame feature extraction and aggregation for action spotting in soccer game broadcasts. Multi-stream networks merge additional optical flow [29, 186] or excitement [15, 171] information to improve game highlights identification. Furthermore, attention models are fed with per-frame semantic information such as pixel information [37] or player localization [108] to extract targeted frame features. Our approach consists to leverage the temporal context information around actions to handle the intrinsic temporal patterns representing these actions.

Deep video understanding models are trained with large-scale datasets. While early works leveraged small custom video sets, a few large-scale datasets are available and worth mentioning, in particular Sports-1M [106] for generic sports video classification, MLB-Youtube [145] for baseball activity recognition, and GolfDB [133] for golf swing sequencing. These datasets all tackle specific tasks in sports. As we focus on soccer in this thesis, we use SoccerNet [72] to assess the performance of our context-aware loss for action spotting.

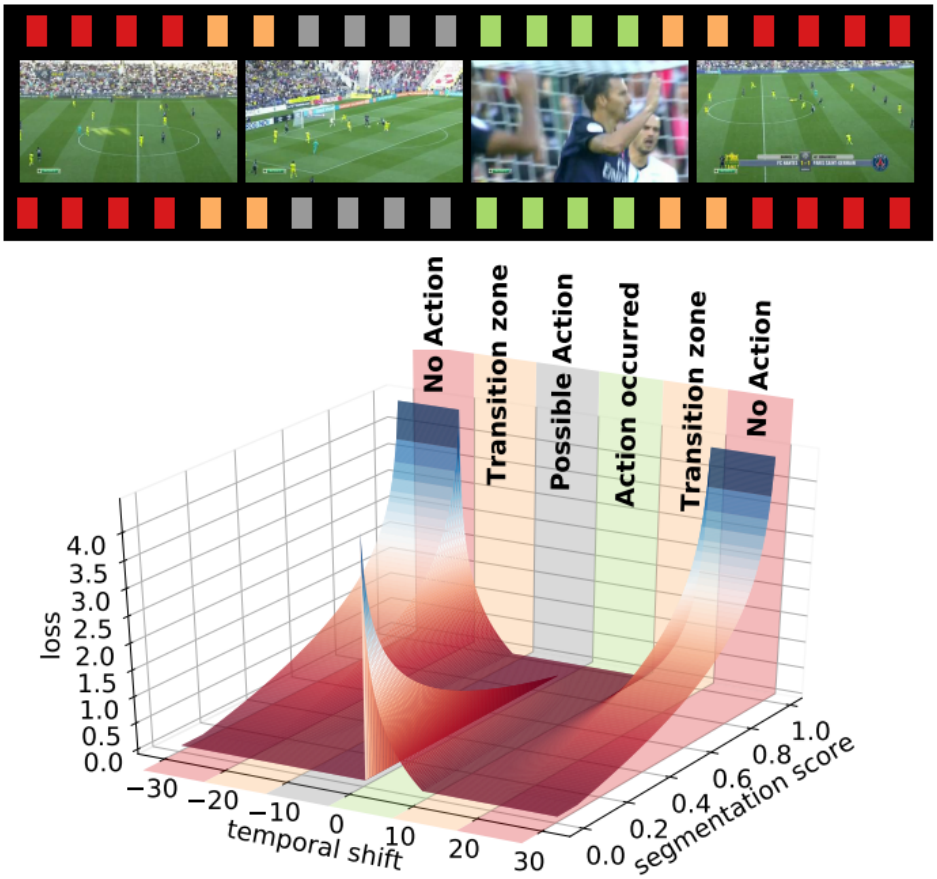


Figure 6.1: Context-aware loss function. We design a novel loss that leverages the temporal context around an action spot (at a temporal shift of 0). We heavily penalize the frames **far-distant** from the action and decrease the penalty for those **gradually closer**. We do not penalize the frames **just before** the action to avoid providing misleading information as its occurrence is uncertain, but we heavily penalize those **just after**, as the action has occurred.

6.2.2 Universal video understanding

Recent video challenges [81] include activity localization, that find temporal boundaries of activities. Following object localization, two-stage approaches have been proposed including proposal generation [26] and classification [25]. SSN [222] models each action instance with a structured temporal pyramid and TURN TAP [69] predicts action proposals and regresses the temporal boundaries, while GTAN [126] dynamically optimizes the temporal scale of each action proposal with Gaussian kernels. BSN [122], MGG [125] and BMN [120] regress the time of activity boundaries, showing state-of-the-art performances on both ActivityNet 1.3 [81] and Thumos'14 [103]. Alternatively, ActionSearch [6] tackles the spotting task iteratively, learning to predict which frame to visit next in order to spot a given activity. However, this method requires sequences of temporal annotations by human annotators to train the models that are not readily available for datasets outside ActivityNet. Also, Alwassel [5] define an action spot as positive as soon as it lands within the boundary of an activity, which is less constraining than the action spotting defined in SoccerNet.

Recently, Sigurdsson [172] question boundaries sharpness and show that human agreement on temporal boundaries reach an average tIoU of 72.5% for Charades [173] and 58.7% on MultiTHUMOS [208]. Alwassel [5] confirm such disparity on ActivityNet [81], but also show that it does not constitute a major roadblock to progress in the field. Different from activity localization, SoccerNet [72] proposes an alternative action spotting task for soccer action understanding, leveraging a well-defined set of soccer rules that define a single temporal anchor per action. We improve the SoccerNet action spotting baseline by introducing a novel context-aware loss that temporally slices the vicinity of the action spots. Also, we integrate our loss for generic activity localization and detection on a boundary-based method [120, 122].

6.3 Context-aware loss function and action spotting network

We address the action spotting task by developing a context-aware loss for a temporal segmentation module, and a YOLO-like loss for an action spotting module that outputs the *spotting predictions* of the network. We first present the re-encoding of the annotations needed for the segmentation and spotting tasks, then we explain how the losses of these modules are computed based on the re-encodings.

We denote by C the number of classes of the action spotting problem. Each action is identified by a single *action frame* annotated as such. Each frame of a given video is annotated with either a one-hot encoded vector with C components for the action frames or a vector of C zeros for the background frames. We denote by N_F the number of frames in a video.

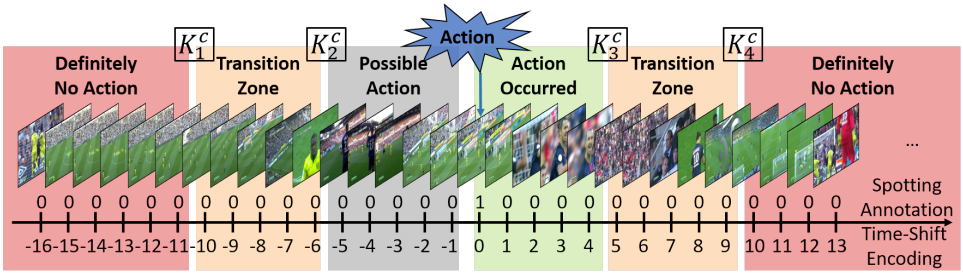


Figure 6.2: Action context slicing. We define six temporal segments around each ground-truth action spot, each of which induces a specific behavior in our context-aware loss function when training the network. **Far before** and **far after** the action, its influence is negligible, thus we train the network not to predict an action. **Just before** the action, we do not influence the network since a particular context may or may not result in an action (*i.e.* an attacking phase *may* lead to a goal). **Just after** the action, its contextual information is rich and unambiguous as the action has just occurred (*i.e.* a goal *leads* to celebrating). Hence, we train the network to predict an action. Finally, we define **transition zones** for our loss function to be smooth, in which we softly train the network not to predict an action. For each class c, the temporal segments are delimited by specific slicing parameters K_i^c and are materialized through our time-shift encoding, which contains richer temporal context information about the action than the initial binary spotting annotation.

6.3.1 Encoding the ground truth

To train our network, the initial annotations are re-encoded in two different ways: with a *time-shift encoding* used for the temporal segmentation loss, and with a *YOLO-like encoding* used for the action spotting loss.

6.3.1.1 Time-shift encoding (TSE) for temporal segmentation

We slice the temporal context around each action into segments related to their distance from the action, as shown in Figure 6.2. The segments regroup frames that are either *far before*, *just before*, *just after*, *far after* an action, or in transition zones between these segments.

We use the segments in our temporal segmentation module so that its *segmentation scores* reflect the following ideas. **(1)** *Far before* an action spot of some class, we cannot foresee its occurrence. Hence, the score for that class should indicate that no action is occurring. **(2)** *Just before* an action, its occurrence is uncertain. Therefore, we do not influence the score towards any particular direction. **(3)** *Just after* an action has happened, plenty of visual cues allow for the detection of the occurrence of the action. The score for its class should reflect the presence of the action. **(4)** *Far after* an action, the score for its class should indicate that it is not occurring anymore. The segments around the actions

of class c are delimited by four temporal *context slicing parameters* $K_1^c < K_2^c < 0 < K_3^c < K_4^c$ as shown in Figure 6.2.

The context slicing is used to perform a *time-shift encoding* (TSE) of each frame x of a video with a vector of length C, containing class-wise information on the relative location of x with respect to its closest past or future actions. The TSE of x for class c, noted $s^c(x)$, is the time-shift (*i.e.* difference in frame indices) of x from either its closest past or future ground-truth action of class c, depending on which has the dominant influence on x. We set $s^c(x)$ as the time-shift from the past action if either **(i)** x is just after the past action; or **(ii)** x is in the transition zone after the past action, but is far before the future action; or **(iii)** x is in the transition zones after the past and before the future actions while being closer to the past action. In all other cases, $s^c(x)$ is the time-shift from the future action.

If x is both located *far after* the past action and *far before* the future action, selecting either of the two time-shifts has the same effect in our loss. Furthermore, for the frames located either before the first or after the last annotated action of class c, only one time-shift can be computed and is thus set as $s^c(x)$. Finally, if no action of class c is present in the video, then we set $s^c(x) = K_1^c$ for all the frames. This induces the same behavior in our loss as if they were all located far before their closest future action. More details are provided in Section C.2.

6.3.1.2 YOLO-like encoding for action spotting

Inspired by YOLO [155], each ground-truth action of the video engenders an *action vector* composed of $2+C$ values. The first value is a binary indicator of the presence ($=1$) of the action. The second value is the location of the frame annotated as the action, computed as the index of that frame divided by N_F. The remaining C values represent the one-hot encoding of the action. We encode a whole video containing N_{GT} actions in a matrix $\mathbf{Y}$ of dimension $N_{\text{GT}} \times (2+C)$, with each line representing an action vector of the video.

6.3.2 Definition of the losses

6.3.2.1 Temporal segmentation loss

The TSE parameterizes the temporal segmentation loss described below. For clarity, we denote by p the segmentation score for a frame x to belong to class c output by the segmentation module, and s as the TSE of x for class c. We detail the loss generated by p in this setting, noted $L(p,s)$. First, in accordance with Figure 6.2, we compute $L(p,s)$ as follows:

$$L(p,s) = \begin{cases} -\ln(1-p) & s \leq K_1^c, \\ -\ln\left(1 - \frac{K_2^c - s}{K_2^c - K_1^c}\, p\right) & K_1^c < s \leq K_2^c, \\ 0 & K_2^c < s < 0, \\ -\ln\left(\frac{s}{K_3^c} + \frac{K_3^c - s}{K_3^c}\, p\right) & 0 \leq s < K_3^c, \\ -\ln\left(1 - \frac{s - K_3^c}{K_4^c - K_3^c}\, p\right) & K_3^c \leq s < K_4^c, \\ -\ln(1-p) & s \geq K_4^c. \end{cases} \tag{6.1}$$

Then, following the practice in [45, 163] to help the network focus on improving its worst segmentation scores, we zero out the loss for scores that are satisfying enough. In the case of Line 4 of Equation 6.1 when $s = 0$, we say that a score is satisfactory when it exceeds some *maximum margin* $\tau_{\max}$. In the cases of Lines 1 and 6 of Equation 6.1, we say that a score is satisfactory when it is lower than some *minimum margin* $\tau_{\min}$. The range of values for p that leads to zeroing out the loss varies with s and the slicing parameters in most cases. This is achieved by revising $L(p,s)$ as in Equation 6.1. Figure 6.1 shows a representation of $\tilde{L}(p,s)$.

$$\tilde{L}(p,s) = \begin{cases} \max\big(0, L(p,s) + \ln(\tau_{\max})\big) & 0 \leq s < K_3^c, \\ \max\big(0, L(p,s) + \ln(1-\tau_{\min})\big) & \text{otherwise.} \end{cases} \tag{6.2}$$

Finally, the segmentation loss $\mathcal{L}^{\text{seg}}$ for a given video of frames $x_1, \ldots, x_{N_F}$ is given in Equation 6.3.

$$\mathcal{L}^{\text{seg}} = \frac{1}{C\, N_F} \sum_{i=1}^{N_F} \sum_{c=1}^{C} \tilde{L}(p^c(x_i), s^c(x_i)). \tag{6.3}$$

6.3.2.2 Action spotting loss

Let N_{pred} be a fixed number of action spotting predictions generated by our network for each video. Those predictions are encoded in $\hat{\mathbf{Y}}$ of dimension $N_{\text{pred}} \times (2 + C)$, similarly to $\mathbf{Y}$.

We leverage an iterative one-to-one matching algorithm to pair each of the N_{GT} ground-truth actions with a prediction. First, we match each ground-truth location of $\mathbf{Y}_{\cdot,2}$ with its closest predicted location in $\hat{\mathbf{Y}}_{\cdot,2}$, and vice-versa (*i.e.* we match the predicted locations with their closest ground-truth locations). Next, we form pairs of (ground-truth, predicted) locations that reciprocally match, we remove them from the process, and we iterate until all ground truths are coupled with a prediction. Consequently, we build $\hat{\mathbf{Y}}^M$ as a reorganized version of the actions encoded in $\hat{\mathbf{Y}}$, such that $\mathbf{Y}_{i,2}$

and $\hat{\mathbf{Y}}^M_{i,2}$ reciprocally match for all $i \leq N_{\text{GT}}$. More details on this matching is provided in Section C.1.

We define the action spotting loss $\mathcal{L}^{\text{as}}$ in Equation 6.4. It corresponds to a weighted sum of the squared errors between the matched predictions and a regularization on the confidence score of the unmatched predictions.

$$\mathcal{L}^{\text{as}} = \sum_{i=1}^{N_{\text{GT}}} \sum_{j=1}^{2+C} \alpha_j \left(\mathbf{Y}_{i,j} - \hat{\mathbf{Y}}^M_{i,j}\right)^2 + \beta \sum_{i=N_{\text{GT}}+1}^{N_{\text{pred}}} \left(\hat{\mathbf{Y}}^M_{i,1}\right)^2. \tag{6.4}$$

6.3.2.3 Complete loss

The final loss $\mathcal{L}$ is presented in Equation 6.5 as a weighted sum of $\mathcal{L}^{\text{seg}}$ and $\mathcal{L}^{\text{as}}$:

$$\mathcal{L} = \lambda^{\text{as}} \mathcal{L}^{\text{as}} + \lambda^{\text{seg}} \mathcal{L}^{\text{seg}}. \tag{6.5}$$

6.3.3 CALFNet: a network for action spotting

The architecture of our network, called *CALFNet*, is illustrated in Figure 6.3 and further detailed in Appendix A.2. We leverage frame feature representations for the videos (*e.g.*, ResNet) provided with the dataset, embodied as the output of the frame feature extractor of Figure 6.3. The temporal CNN of Figure 6.3 is composed of a spatial two-layer MLP, followed by four multi-scale 3D convolutions (*i.e.* across time, features and classes). The temporal CNN outputs a set of $C \times f$ features for each frame organized in C feature vectors (one per class) of size f, as in [163]. These features are input into a segmentation module, in which we use Batch Normalization [95] and sigmoid activations. The closeness of the C vectors obtained in this way to a pre-defined vector gives the C segmentation scores output by the segmentation module, as [45]. The $C \times f$ features obtained previously are concatenated with the C scores and fed to the action spotting module, as shown in Figure 6.3. It is composed of three successive temporal max-pooling and 3D convolutions, and outputs N_{pred} vectors of dimension $(2+C)$. The first two elements of these vectors are sigmoid-activated, the C last are softmax-activated. The activated vectors are stacked to produce the prediction $\hat{\mathbf{Y}}$ of dimension $N_{\text{pred}} \times (2+C)$ for the action spotting task.

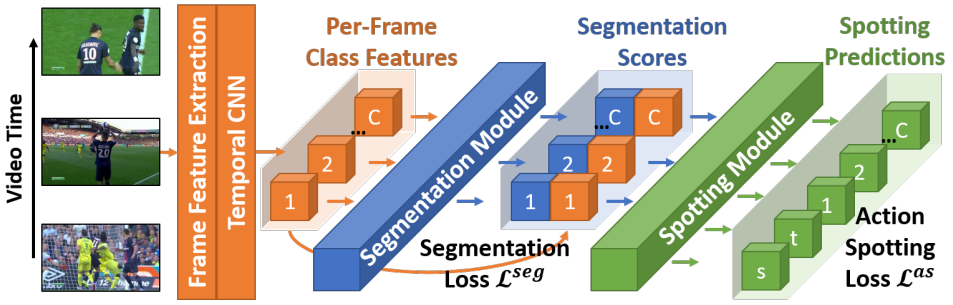


Figure 6.3: Our network for action spotting. We propose a novel network architecture, called *CALFNet*, made of a **frame feature extractor** and a **temporal CNN** outputting C class feature vectors per frame, a **segmentation module** outputting per-class segmentation scores, and a **spotting module** extracting $2+C$ values per spotting prediction (*i.e.* the confidence score s for the spotting, its location t and a per-class prediction).

6.4 Experiments

6.4.1 Experiments on SoccerNet

6.4.1.1 Experimental setup

Three classes of action are annotated in SoccerNet by Giancola *et al.* [72]: goals, cards, and substitutions, so $C = 3$ in this case. They identify each action by one annotated frame: the moment the ball crosses the line for *goal*, the moment the referee shows a player a card for *card*, and the moment a new player enters the field for *substitution*. We train our network on the frame features already provided with the dataset. Giancola *et al.* first subsampled the raw videos at 2 fps, then they extracted the features with a backbone network and reduced them by PCA to 512 features for each frame of the subsampled videos. Three sets of features are provided, each extracted with a particular backbone network: I3D [31], C3D [185], and ResNet [80].

We measure performances with the action spotting metric introduced in SoccerNet [72]. An action spot is defined as positive if its temporal offset from its closest ground truth is less than a given tolerance δ. The average precision (AP) is estimated based on Precision-Recall curves, then averaged between classes (mAP). An Average-mAP is proposed as the AUC of the mAP over different tolerances δ ranging from 5 to 60 seconds. This dataset and the evaluation metric are further detailed in Section 7.2.

We train our network on batches of *chunks*. We define a chunk as a set of N_F contiguous frame feature vectors. We set $N_F = 240$ to maintain a high training speed while retaining sufficient contextual information. This size corresponds to a clip of 2 minutes

of raw video. A batch contains chunks extracted from a single raw video. We extract a chunk around each ground-truth action, such that the action is randomly located within the chunk. Then, to balance the batch, we randomly extract N_{GT}/C chunks composed of background frames only. An epoch ends when the network has been trained on one batch per training video. At each epoch, new batches are re-computed for each video for data augmentation purposes. Each raw video is time-shift encoded before training. Each new training chunk is encoded with the YOLO-like encoding.

The number of action spotting predictions generated by the network is set to $N_{pred} = 5$, as we observed that no chunks of 2 minutes of raw video contain more than 5 actions. We train the network during 1000 epochs, with an initial learning rate $lr = 10^{-3}$ linearly decreasing to 10^{-6}. We use the Adam optimizer with default parameters [111].

For the segmentation loss, we set the margins $\tau_{max} = 0.9$ and $\tau_{min} = 0.1$ in Equation 6.2, following the practice in [163]. For the action spotting loss in Equation 6.4, we set $\alpha_j = 1$ for $j \neq 2$, while α_2 is optimized (see below) to find an appropriate weighting for the location components of the predictions. Similarly, β is optimized to find the balance between the loss of the action vectors and the regularization of the remaining predictions. For the final loss in Equation 6.5, we optimize λ^{seg} to find the best balance between the two losses.

6.4.1.2 Hyperparameter optimization

For each set of features (I3D, C3D, ResNet), we perform a joint Bayesian optimization [2] on the number of frame features f extracted per class, on the temporal receptive field r of the network (*i.e.* temporal kernel dimension of the 3D convolutions), and on the parameters $\alpha_2, \beta, \lambda^{seg}$. Next, we perform a grid search optimization on the slicing parameters K_i^c.

For ResNet, we obtain $f = 16$, $r = 80$, $\alpha_2 = 5$, $\beta = 0.5$, $\lambda^{seg} = 1.5$. For goals (resp. cards, substitutions) we have $K_1 = -40$ (resp. -40, -80), $K_2 = -20$ (resp. -20, -40), $K_3 = 120$ (resp. 20, 20), and $K_4 = 180$ (resp. 40, 40). Given the framerate of 2 fps, those values can be translated to seconds by scaling them down by a factor of 2. The value $r = 80$ corresponds to a temporal receptive field of 20 seconds on both sides of the central frame in the temporal dimension of the 3D convolutions.

6.4.1.3 Main results

The performances obtained with the optimized parameters are reported in Table 6.1. As shown, we establish a new state-of-the-art performance on the action spotting task of SoccerNet, outperforming the previous benchmark by a comfortable margin, for all the frame features. ResNet gives the best performance, as also observed in [72]. A sensitivity analysis of the parameters K_i^c reveals robust performances around the optimal values, in-

dicating that no heavy fine-tuning is required for the context slicing. Also, performances largely decrease as the slicing is strongly reduced, which emphasizes its usefulness.

Method	Frame features		
	I3D	C3D	ResNet
SoccerNet baseline 5s [72]	-	-	34.5
SoccerNet baseline 5s [72]	-	-	40.6
SoccerNet baseline 5s [72]	-	-	49.7
Vanderplatse *et al.* [192] (+audio features)			56.0
Vats *et al.* [193]	-	-	60.1
CALFNet (Ours)	53.6	57.7	62.5

Table 6.1: Results on SoccerNet. Average-mAP (in %) on the test set of SoccerNet for the action spotting task. We establish a new state-of-the-art performance.

6.4.1.4 Ablation study

Since the ResNet features provide the best performance, we use them with their optimized parameters for the following ablation studies. **(i)** We remove the segmentation module, which is equivalent to setting $\lambda^{\text{seg}} = 0$ in Equation 6.5. This also removes the context slicing and the margins τ_{max} and τ_{min}. **(ii)** We remove the action context slicing such that the ground truth for the segmentation module is the raw binary annotations, *i.e.* all the frames must be classified as background except the action frames. This is equivalent to setting $K_1 = -1 = K_2 = -K_3 = -K_4$. **(iii)** We remove the margins that help the network focus on improving its worst segmentation scores, by setting $\tau_{\text{max}} = 1$, $\tau_{\text{min}} = 0$ in Equations 6.2. **(iv)** We remove the iterative one-to-one matching between the ground truth $\mathbf{Y}$ and the predictions $\hat{\mathbf{Y}}$ before the action spotting loss, which is equivalent to using $\hat{\mathbf{Y}}$ instead of $\hat{\mathbf{Y}}^M$ in Equation 6.4. The results of the ablation studies are shown in Table 6.2.

From an Average-mAP perspective, the auxiliary task of temporal segmentation improves the performance on the action spotting task (from 58.9% to 62.5%), which is a common observation in multi-task learning [213]. When the segmentation is performed, our temporal context slicing gives a significant boost compared to using the raw binary annotations (from 57.8% to 62.5%). This observation is in accordance with the sensitivity analysis. It also appears that it is preferable to not use the segmentation at all rather than using the segmentation with the raw binary annotations (58.9% vs 57.8%), which further underlines the usefulness of the context slicing. A boost in performance is also observed when we use the margins to help the network focus on improving its worst segmentation scores (from 59.0% to 62.5%). Eventually, Table 6.2 shows that it is extremely beneficial to match the predictions of the network with the ground truth before the action spotting loss (from 46.8% to 62.5%). This makes sense since there is no point in evaluating the network on its ability to order its predictions, which is a hard and unnecessary con-

straint. The large impact of the matching is also justified by its direct implication in the action spotting task assessed through the Average-mAP.

	Segmentation	Slicing	Margins	Matching	Results
(i)				✓	58.9
(ii)	✓		✓	✓	57.8
(iii)	✓	✓		✓	59.0
(iv)	✓	✓	✓		46.8
Ours	✓	✓	✓	✓	**62.5**

Table 6.2: Ablation study. We perform ablations by **(i)** removing the segmentation ($\lambda^{\text{seg}} = 0$), hence the slicing and the margins; **(ii)** removing the context slicing ($K_1 = -1 = K_2 = -K_3 = -K_4$); **(iii)** removing the margins that help the network focus on improving its worst segmentation scores ($\tau_{\min} = 0, \tau_{\max} = 1$); **(iv)** removing the matching (using $\hat{\mathbf{Y}}$ instead of $\hat{\mathbf{Y}}^M$ in $\mathcal{L}^{\text{as}}$). Each part evidently contributes to the overall performance.

6.4.1.5 Results through game time

In soccer, it makes sense to analyze the performance of our model through game time, since the actions are not uniformly distributed throughout the game. For example, a substitution is more likely to occur during the second half of a game. We consider non-overlapping bins corresponding to 5 minutes of game time and compute the Average-mAP for each bin. Figure 6.4 shows the evolution of this metric through game time.

It appears that actions occurring during the first five minutes of a half-time are substantially more difficult to spot than the others. This may be partially explained by the occurrence of some of these actions at the very beginning of a half-time, for which the temporal receptive field of the network requires the chunk to be temporally padded. Hence, some information may be missing to allow the network to spot those actions. Besides, when substitutions occur during the break, they are annotated as such on the first frame of the second halves of the matches, which makes them practically impossible to spot. In the test set, this happens for 28% of the matches. None of these substitutions are spotted by our model, which thus degrades the performances during the first minutes of play in the second halves of the matches. However, they merely represent 5% of all the substitutions, and removing them from the evaluation only boosts our Average-mAP by 0.7% (from 62.5% to 63.2%).

6.4.1.6 Results as a function of vicinity

We investigate whether actions are harder to spot when they are close to each other. We bin the ground-truth actions based on the distance that separates them from the previous (or next, depending on which is the closest) ground-truth action, regardless of their

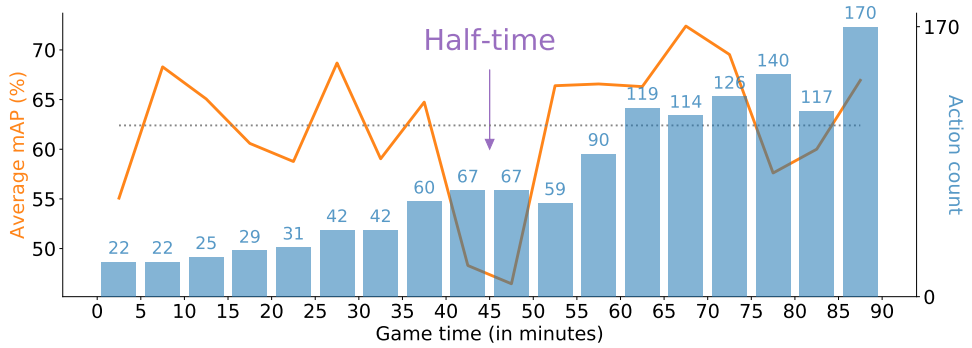


Figure 6.4: Performance as function of game time. **Average-mAP** spotting performance over the game time with all ground-truth actions of the dataset binned in 5 minute intervals. It appears that actions around the half-time break are more challenging to spot. **Number of actions** for each bin. **Our performance** (62.5%).

classes. Then, we compute the Average-mAP for each bin. The results are represented in Figure 6.5.

We observe that the actions are more difficult to spot when they are close to each other. This could be due to the reduced number of visual cues, such as replays, when an action occurs rapidly after another and thus must be broadcast. Some confusion may also arise because the replays of the first action can still be shown after the second action, *e.g.*, a sanctioned foul followed by a converted penalty. This analysis also shows that the action spotting problem is challenging even when the actions are further apart, as the performances in Figure 6.5 eventually plateau.

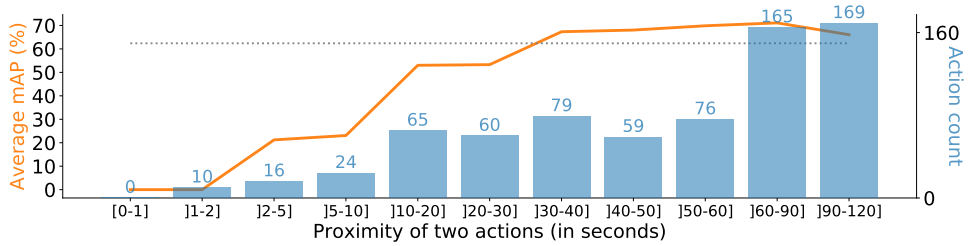


Figure 6.5: Performance as function of action vicinity. **Average-mAP** spotting performance per bin of ground-truth actions grouped by distance (in seconds) from their closest ground-truth action. It appears that nearby actions are more challenging to spot. **Number of actions** for each bin. **Our performance** (62.5%).

6.4.1.7 Per class results

We perform a per-class analysis in a similar spirit as the Average-mAP metric. For a given class, we fix a tolerance δ around each annotated action to determine positive predictions and we aggregate these results in a confusion matrix. An action is considered spotted when its confidence score exceeds some threshold optimized for the F_1 score on the validation set. From the confusion matrix, we compute the precision, recall and F_1 score for that class and for that tolerance δ. Varying δ from 5 to 60 seconds provides the evolution of the three metrics as a function of the tolerance. Figure 6.6 shows these curves for *goals* for our model and for the baseline [72]. The results for cards and substitutions are provided in Appendix C.3.1.

Figure 6.6 shows that most goals can be efficiently spotted by our model within 10 seconds around the ground truth (δ = 20 seconds). We achieve a precision of 80% for that tolerance. The previous baseline plateaus within 20 seconds (δ = 40 seconds) and still has a lower performance. In particular for goals, many visual cues facilitate their spotting, *e.g.*, multiple replays, particular camera views, or celebrations from the players and from the public.

6.4.2 Experiments on ActivityNet

In this section, we evaluate our context-aware loss in a more generic task than action spotting in soccer videos. We tackle the *Activity Proposal* and *Activity Detection* tasks of the challenging ActivityNet dataset, for which we use the ResNet features provided with the dataset at 5 fps.

6.4.2.1 Experimental setup

We use the current state-of-the-art network, namely BMN [120], with the code provided in [3]. BMN is equipped with a temporal evaluation module (TEM), which plays a similar role as our temporal segmentation module. We replace the loss associated with the TEM by our novel temporal segmentation loss $\mathcal{L}^{\text{seg}}$. The slicing parameters are set identically for all the classes and are optimized with respect to the AUC performance on the validation set by grid search with the constraint $K_1 = 2K_2 = -2K_3 = -K_4$. The optimization yields to the best results when $K_1 = -14$.

6.4.2.2 Main results

The average performances on 20 runs of our experiment and of the BMN base code [3] are reported in Table 6.3. Our novel temporal segmentation loss improves the performance obtained with BMN [3] by 0.15% and 0.12% for the activity proposal task (AR@100

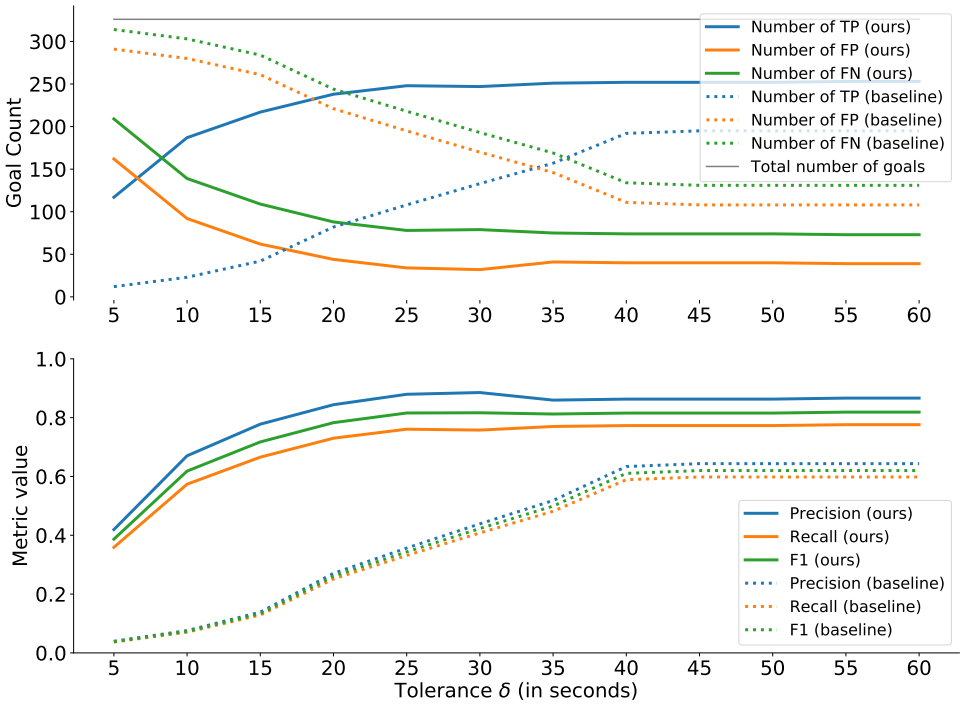


Figure 6.6: Per-class results (goals). A prediction of class *goal* is a **true positive (TP)** with tolerance δ when it is located at most $\delta/2$ seconds from a ground-truth goal. The baseline results are obtained from the best model of [72]. Our model spots most goals within 10 seconds around the ground truth ($\delta = 20$ seconds).

and AUC) and by 0.38% for the activity detection task (Average-mAP). These increases compare with some recent increments, while being obtained just by replacing their TEM loss by our context-aware segmentation loss. The network thus has the same architecture and number of parameters. We conjecture that our loss $\mathcal{L}^{\text{seg}}$, through its particular context slicing, helps train the network by modeling the uncertainty surrounding the annotations. Indeed, it has been shown in [5, 172] that a large variability exists among human annotators on which frames to annotate as the beginning and the end of the activities of the dataset. Let us note that in BMN, the TEM loss is somehow adapted around the action frames in order to mitigate the penalization attributed to their neighboring frames. Our method goes one step further, by directly designing a temporal context-aware segmentation loss.

Method	AR@100	AUC	Average-mAP
Lin *et al.* [121]	73.01	64.40	29.17
Gao *et al.* [68]	73.17	65.72	-
BSN [122]	74.16	66.17	30.03
P-GCN [216]	-	-	31.11
BMN [120]	75.01	67.10	**33.85**
BMN code [3]	75.11	67.16	30.67 ± 0.08
Ours: [3] + $\mathcal{L}^{seg}$	**75.26**	**67.28**	31.05 ± 0.07

Table 6.3: Results on ActivityNet validation set for the proposal task (AR@100, AUC) and for the detection task (Average-mAP). For our experiments, we report the average values on 20 runs.

6.5 Automatic highlight generation

Some action spotting and temporal segmentation results are shown in Figure 6.7. It appears that some sequences of play have a high segmentation score for some classes but do not lead, quite rightly, to an action spotting. It turns out that these sequences are often related to unannotated actions of supplementary classes that resemble those considered so far, such as *unconverted goal opportunities* and *unsanctioned fouls*.

To quantify the spotting results of goal opportunities, we can only compute the precision metric since these actions are not annotated. We manually inspect each video sequence of the test set where the segmentation score for goals exceeds some threshold η but where no ground-truth goal is present. We decide whether the sequence is a goal opportunity or not by asking two frequent observers of soccer games if they would include it in the highlights of the match. The sequence is a true positive when they both agree to include it and a false positive, otherwise. The precision is then computed for that η. By gradually decreasing η from 0.9 to 0.3, we obtain the precision curve shown in Figure 6.8. It appears that 80% of the sequences with a segmentation score larger than $\eta = 0.5$ are considered goal opportunities.

As a direct by-product, we derive an automatic highlights generator without explicit supervision. We extract a video clip starting 15 seconds before each spotting of a *goal* or a *card* and ending 20 seconds after. We proceed likewise for the sequences with a segmentation score ≥ 0.5 for *goals*. We dismiss substitutions as they rarely appear in highlights. We assemble the clips chronologically to produce the highlights video. Evaluating its quality is subjective, but we found its content to be adequate, even if the montage could be improved. Indeed, only sequences where a goal, a goal opportunity, or a foul occurs are selected. This reinforces the usefulness of the segmentation, as it provides a direct overview of the proceedings of the match, including proposals for unannotated actions that are interesting for highlights.

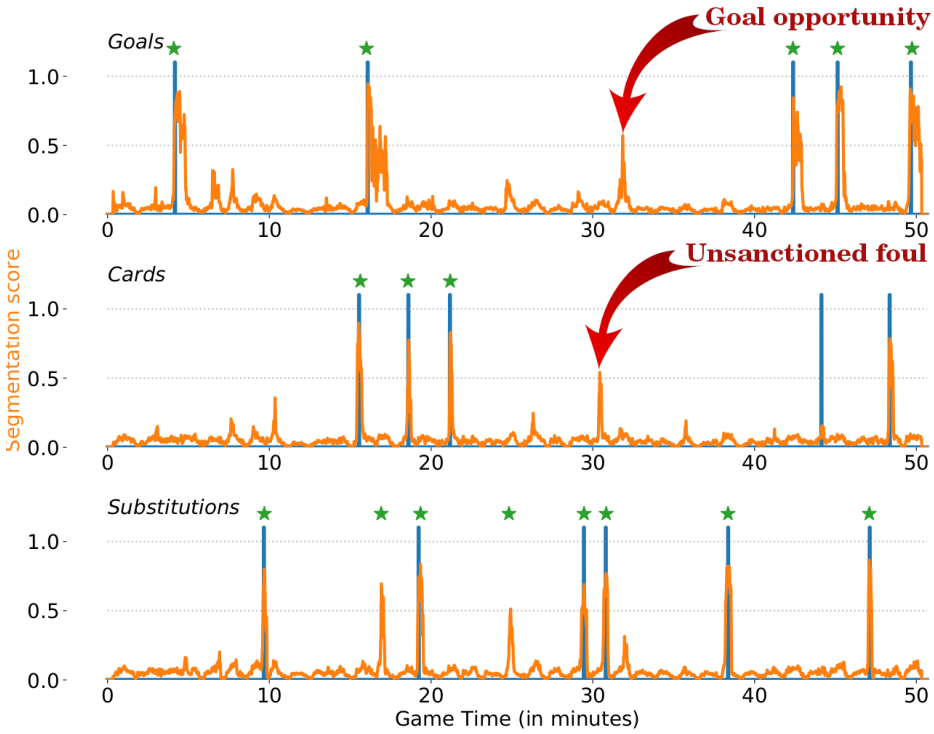


Figure 6.7: Action spotting and segmentation for the 2nd half of the *Remuntada* FCB - PSG. **Ground truth actions**, **temporal segmentation curves**, and **spotting results** are illustrated. We can identify **unannotated interesting actions** using our segmentation.

6.6 Conclusion

In this chapter, we have tackled the challenging action spotting task of SoccerNet with a novel context-aware loss for segmentation and a YOLO-like loss for the spotting. The former treats the frames according to their time-shift from their closest ground-truth actions. The latter leverages an iterative matching algorithm that alleviates the need for the network to order its predictions. To show generalization capabilities, we also test our context-aware loss on ActivityNet. We improve the state-of-the-art on ActivityNet by 0.15% in AR@100, 0.12% in AUC, and 0.38% in Average-mAP, by only including our context-aware loss without changing the network architecture. We achieve a new state-of-the art on SoccerNet, surpassing by far the previous baseline (from 49.7% to 62.5% in Average-mAP) and spotting most actions within 10 seconds around their ground truth. Both our context-aware loss and matching algorithm are shown to be key components in this performance. Finally, we leverage the resulting segmentation results to identify

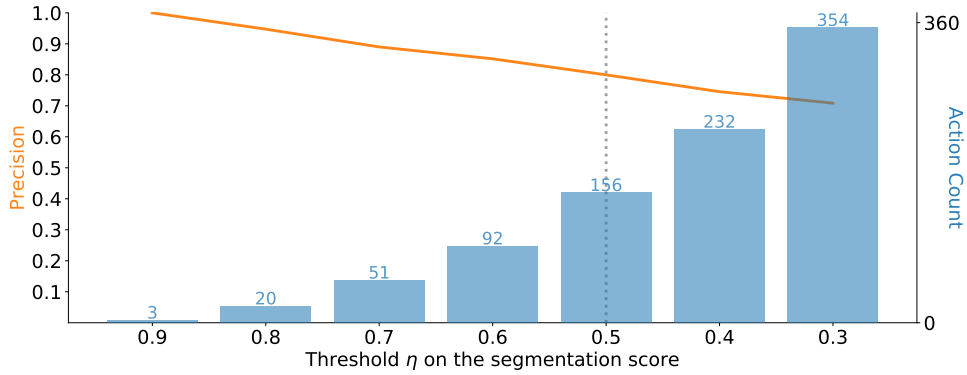


Figure 6.8: Precision for goal opportunities, as a function of the threshold on the segmentation score to exceed for manually inspecting a sequence. For scores larger than $\eta = 0.5$, a **precision** of 0.8 is achieved, *i.e.* 80% of the sequences inspected were goal opportunities. **Number of sequences** inspected per threshold.

unannotated actions such as goal opportunities and derive a highlights generator without specific supervision.

By doing so, we showed another approach than the one of the previous chapter to extract high-level semantics about the game and proposed to solve a real-world application with high-level semantics. In the next chapter, we broaden the set of high-level semantics by providing new annotations that augment the SoccerNet dataset. This opens the path for many soccer-related applications, such as the automatic production of live events.

CHAPTER 7

Towards a broader set of high-level semantics in soccer videos

Contents

In the two previous chapters, we showed how high-level semantics such as game phases (see Chapter 5) and game events (see Chapter 6) could be extracted from soccer videos. Even though our methods perform well on their respective task, the variety of extracted semantics can still be considered as too limitative for a complete interpretation of the soccer game, as only 4 types of game phases and 3 classes of game events were considered. Hence, we take a first step towards a broader spectrum of obtainable high-level semantics by proposing a substantial extension of the SoccerNet action spotting dataset.

As a first contribution, we enlarge the panel of game events to 17 classes instead of 3, including all game events that typically occur during a soccer game. Furthermore, we provide an additional annotation for each action spot related to whether or not the action is explicitly shown in the TV broadcast. By doing so, we are able to evaluate the spotting performances on unshown actions as well.

Next, we provide new types of semantics that are directly related to the producer's job, whose role is to properly combine different cameras to convey emotions to the viewer, while ensuring an easy understanding of the game. To do so, we start by annotating the link between each camera shot of a replay and its corresponding live action within the game. Furthermore, for a subset of these games, we also annotate the class of each camera shot as well as its temporal boundaries. This gathered knowledge on how a TV broad-

cast is edited brings valuable semantics for applications such as automatic production of live events or highlights generation.

This chapter is organized as follows. In Section 7.1, we present some well-known datasets for video understanding and motivate the need for dataset with more annotations. Then, the original SoccerNet dataset is described in Section 7.2. In Section 7.3, we present our novel annotations and suggest different semantic tasks around them. Next, we show that our network, developed in Chapter 6, also proves to be effective for the action spotting task on the new classes in Section 7.4. Finally, Section 7.5 concludes on this work and proposes some possible future works based on these additional annotations.

CONTRIBUTIONS RELATED TO THIS CHAPTER

Contributions. **(i)** We present a proposal to expand the SoccerNet dataset by extending its action spotting annotations to 17 classes and by providing new annotations related to the editing of the TV broadcast; these additions will be part of the coming new SoccerNet-v2 dataset, **(ii)** We propose an extended action spotting task, based on unshown action spotting, involving the context rather than the visual information. **(iii)** We define several tasks related to the editing of the TV broadcast. **(iv)** We propose a new task that aims at linking the replays to their corresponding live action spot.

7.1 Introduction

Supervised video understanding has recently witnessed an increase in interest thanks to the emergence of large annotated datasets, focusing on different types of high-level semantics such as activity understanding [120, 122, 216], video alignment [10, 77], or video captioning [91, 92, 151, 226].

ActivityNet [81] stands as one of the most popular datasets for video understanding. Each year, many researchers compete to reach top performances on the different tasks of this dataset, *e.g.*, *temporal activity detection*, where the objective is to predict the class and the extent of activities presented in a video with "start" and "end" boundaries, and *dense-captioning events*, aiming to detect and describe textually the events occurring in a video. Similarly, the AVA-Kinetics dataset [116] combines the AVA Actions dataset [74], specialized in spatio-temporal action localization, and the Kinetics-700 dataset [30], covering human action classes, to provide *localized human action annotations*. The proposed task is particularly challenging as it aims at detecting and understanding individual human activities in a universal context, *i.e.* covering a large variety of scenes and activities.

Another interesting dataset for video understanding is the EPIC-Kitchens dataset [43, 44] composed of 55 hours of densely labeled videos with action segments and object bounding boxes in scene-specific kitchen environments. Recently, the authors released

a second version with 100 hours of videos, showing the growing interest of the community for this dataset, comprising several tasks such as *action recognition*, *action detection*, and *action anticipation*, the latter focusing on predicting the next action label given the previous action segment. Even though the videos are restricted to the kitchen environment, the dataset remains challenging since it contains a large variety of human-object interactions.

The sports research community can also rely on video datasets. For example, UCF Sports [174] contains sequences from various sports and aims at recognizing and localizing different actions in these sequences. Sports Videos in the Wild [164] is another sport-centric dataset including 4100 videos representing 30 different sports and 44 classes of actions. It is designed to tackle several tasks such as action recognition, action detection and spatio-temporal alignment. Since this dataset is composed of smartphone footages rather than professional broadcasts, the videos contain jittering, unexpected motions, different view points and occlusions, which makes the setup challenging.

In the context of soccer videos, SoccerNet [72] introduces the task of action spotting, which consists in detecting the actions in a soccer game, where each action is defined by a single action spot. It places itself as the largest and most challenging public dataset for that task. This dataset, which is the core of our extensions, is presented in details in Section 7.2. Recently, another extension of SoccerNet emerged, called SoccerDB [102]. Particularly, this dataset introduces 7 additional action classes and proposes bounding box annotations for the players. The authors also propose a novel *replay detection* task. Their dataset is composed of about half of SoccerNet's videos to which they added 76 matches from the Chinese Super League and from the 2018 World Cup. Even though this dataset brings more annotations to SoccerNet, we believe that it is only a first step towards a more complete understanding of soccer. In fact, it still misses a lot of high-level semantics, such as a complete set of possible actions in soccer and a deeper understanding of the production of TV broadcasts.

In this chapter, we propose to extend SoccerNet even more than SoccerDB, by first increasing the number of action classes to 17, covering all possible actions that usually occur during a soccer game. We also distinguish between actions that are visible on the TV broadcast from the ones that are not shown. In fact, unshown actions can still be derived from the context by humans, but it is a particularly challenging task to automate with algorithms as it requires a higher level of understanding of soccer. Second, we provide annotations related to the production of the TV broadcast, which we call *editing semantics*. Particularly, we provide annotations on the camera shots such as the type of camera that is filming the game among 13 classes (*e.g.*, the main, close-up, or behind the goal camera), its time boundaries and its *replay* or *live* characteristic. By doing so, we introduce the tasks of *camera shot temporal segmentation* and *camera shot boundary detection* as well as a new *replay grounding* task, whose objective is to link each replay to its corresponding live action within the game.

7.2 Original SoccerNet dataset

In this section, we present the original SoccerNet dataset as designed by Giancola *et al*. [72]. First, we describe the data format and the provided annotations in Section 7.2.1. Then, Section 7.2.2 provides details on the action spotting task and its evaluation metric as originally proposed by the authors. Finally, we briefly describe the limitations of the dataset in Section 7.2.3, which motivates the need for our extensions.

7.2.1 Description of the data

7.2.1.1 Input data

As stated in its original paper title, SoccerNet is a scalable dataset for action spotting in soccer video. It is composed of 500 complete soccer games taken from the most famous European soccer leagues: La Liga (Spain), Ligue 1 (France), Lega Serie A (Italy), Bundesliga (Germany), Premier League (England), and the Champions League (Europe), from seasons 2014 to 2017. Each game comprises two videos, one for each half time, corresponding to a broadcast as transmitted on television. Therefore, the dataset is composed of 1000 videos, adding up to about 764 hours of content. Since these videos correspond to TV broadcasts, they consist of an editing of several cameras into a single video flux, with incrusted data such as game time, the score and other sporadic information about the game. These videos were collected using the YouTube platform and originate from different national TV channels. Therefore, the commentators soundtracks and the inscriptions on screen come in different languages.

These videos are provided in two different quality formats. The first one corresponds to the raw videos as extracted from YouTube, mostly in HD resolutions (1280×720), with slight differences between the framerates of the videos as they come from different sources. The second one is a low-quality resolution (398×224) with fixed frame rate of 25 fps, converted from the raw YouTube videos. The problem when working with videos is the overall size of the data. For the sake of comparison, we compute the corresponding size of all videos for both resolutions. The high-resolution videos add up to about 145 TB of raw uncompressed data, which is not tractable for most hardwares. The low-resolution videos still amount to about 16 TB of raw data, which is more suitable, but still too large to deal with in our case.

To alleviate this issue of colossal memory load, the authors of SoccerNet provide features extracted from the videos at 2 fps. To do so, they choose three networks pre-trained on the ImageNet [49] dataset, namely ResNet [80], I3D [31] and C3D [185], which are state-of-the-art architectures for image and video classification. The authors pre-compute all features by extracting the representation of a particular feature map inside of this network for each frame at 2 fps and vectorize it. Therefore, they end up with a smaller 2D matrix per video of dimension (F, N_I), where F is the number of frames in the

2 fps video and N_I the number of features per frame. These features reduce the memory load to only about 40 GB, which is now acceptable for processing. They also propose to further reduce these features to 512 using a custom PCA reduction. Doing so reduces the memory load even more to about 10 GB of data. Therefore, we choose to use these features rather than the raw videos, which would otherwise make the task intractable in practice.

Finally, the games are separated into a *train*, *validation* and *test* set with respectively 300, 100 and 100 games. The split is done randomly and fixed once and for all, at the game level.

7.2.1.2 Labels of action spots

The 500 games presented in the previous section are provided along with some game event annotations, corresponding to one timestamp per game event. Three classes of actions are annotated: goals, cards and substitutions, and their corresponding timestamps are defined as follows:

DEFINITION OF THE ACTION CLASSES

1. *Goal:* the exact moment at which the ball crosses the goal line.
2. *Card:* the exact moment at which the referee shows the card to the player, whether it is a yellow or red card.
3. *Substitution:* the exact moment at which the entering player enters the field by one of the outer field line to replace one of his teammates.

These annotations were collected automatically using a sports website[1] that lists such events with a one-minute precision. They were fine-tuned later on manually to a one second precision, which is more practical for training networks. In total, the dataset contains up to 6637 timestamps corresponding to the three aforementioned classes. Therefore, these annotations can be considered as very sparse as there is only one event occurring every 6.9 minutes on average. This is challenging for detection algorithms as they have to refrain from predicting false action spots most of the time, but still have to be accurate when an action really occurs.

7.2.2 The action spotting task and its evaluation metric

Alongside the data, SoccerNet introduced the novel task of *action spotting*, which consists in finding the exact timestamp at which each action occurs. This task differs from

[1] www.flashscore.info

the temporal activity detection of ActivityNet presented in Section 7.1. In fact, activities are defined with a beginning and an end rather than a single timestamp. This is not adequate in the case of soccer as the boundaries of soccer game events would be quite blurry in most cases (*e.g.*, when does a goal start or end?). This is why the authors chose to define game events with single spots.

With every new task comes an evaluation methodology. The authors of SoccerNet define a novel metric, called the *Average Mean Average Precision* (average-mAP), which shares some similarities with popular evaluation metrics for object detection. This metric is computed as follows.

First, tolerances δ are defined around each ground-truth action spots, where a predicted action spot is considered as correct if it is comprised within this time interval of size δ around a ground-truth action spot. Then, for several tolerances δ (from 5 to 60 seconds with steps of 5 seconds) and several thresholds τ_{conf} on the confidence score of the prediction (200 points evenly spaced between 0 and 1, where only the predictions with a confidence larger than this threshold are considered), three performance indicators are computed for each class c separately on the entire set:

1. the number of true positives $\mathrm{TP}(\delta, \tau_{conf}, c)$, corresponding to predicted action spots that fall within the δ tolerance around unmatched ground-truth action spots,
2. the number of false positives $\mathrm{FP}(\delta, \tau_{conf}, c)$, corresponding either to action spots outside of any tolerance around ground-truth action spots or additional predicted action spots around an already matched ground-truth action spot,
3. the number of false negatives $\mathrm{FN}(\delta, \tau_{conf}, c)$, corresponding to unmatched ground-truth action spots.

Then, for each τ_{conf}, these indicators are projected as a single point in the Precision-Recall space by computing:

$$\mathrm{P}(\delta, \tau_{conf}, c) = \frac{\mathrm{TP}(\delta, \tau_{conf}, c)}{\mathrm{TP}(\delta, \tau_{conf}, c) + \mathrm{FP}(\delta, \tau_{conf}, c)} \tag{7.1}$$

and

$$\mathrm{R}(\delta, \tau_{conf}, c) = \frac{\mathrm{TP}(\delta, \tau_{conf}, c)}{\mathrm{TP}(\delta, \tau_{conf}, c) + \mathrm{FN}(\delta, \tau_{conf}, c)}. \tag{7.2}$$

The pair of coordinates, of the form,

$$\mathrm{PR}(\delta, c) = \{(\mathrm{R}(\delta, \tau_{conf}, c), \mathrm{P}(\delta, \tau_{conf}, c)) \,|\, \tau_{conf} \in \{0, ..., 1\}\} \tag{7.3}$$

then defines a curve in the Precision-Recall space. The area under this curve is then approximated by an 11-point approximation, as defined by the PASCAL VOC dataset [55]. The approximation works as follows: the recall axis is separated into 11 points between 0 and 1 evenly spaced $\{0, 0.1, 0.2, ..., 1\}$. Each of these 11 recall values are then assigned with a precision value corresponding to the maximal precision value amongst the set of points $\text{PR}(\delta, c)$ whose recall is at least greater than its value. This gives us 11 points defined as $\text{PR}_{11}(\delta, c) = \{\text{P}_i(\delta, c) \mid i \in \{0, 1, ..., 11\}$.

Then, an average of these values is computed over the i index. This new metric corresponds to the Average Precision per class $\text{AP}(\delta, c)$ as usually used in object detection. This Average Precision per class is then averaged over all classes c, which corresponds to the Mean Average Precision mAP. Finally, the area under the curve of the $\text{mAP}(\delta)$, as a function of δ, is computed using a trapezoidal integer approximation and divided by the entire area. This gives the final Average Mean Average Precision (average-mAP) metric.

Even though this metric is difficult to interpret due to its non-linear computation of the average precision, it is actually suited for the task of action spotting as it provides a good approximation of the performance of the algorithms on several δ tolerances. Since it removes the dependance on the confidence score by the Average Precision computation, it can fairly compare algorithms with different strategies of confidence score assignments.

7.2.3 Limitations of the dataset

Even though SoccerNet has known a well-deserved success in the scientific community, the dataset presents several limitations:

LIMITATIONS OF SOCCERNET

1. It only contains 3 classes (goals, cards and substitutions), which might be considered as too limitative for soccer applications needing a more complete understanding of the game events (*e.g.*, corners, free kicks, fouls, ...).
2. It only proposes the task of action spotting, while other categories of high-level semantics could be valuable for applications based on soccer videos (*e.g.*, automatic highlights generation, automatic live production, ...).
3. The action spotting semantics only relates to the *content* of the soccer game and does not consider the TV broadcast choices made by the producer (*editing semantics*) which is valuable for several applications such as automatic highlight generation or the automatic production of TV broadcasts.

Our proposals for extending the SoccerNet dataset, which will be part of the new SoccerNet-v2 dataset, circumvent these limitations.

7.3 Extending SoccerNet with new high-level semantics

In this section, we detail our proposals for extensions. As a first extension, we broaden the set of action classes to spot by annotating manually the entire dataset with 17 classes instead of 3, covering all common actions that typically occur during a game. The definitions of these new classes are presented in Section 7.3.1. Then, in Section 7.3.2, we present new annotations related to editing semantics, for which we propose two interesting tasks, namely *camera shot temporal segmentation* and *camera shot boundary detection*. Finally, in Section 7.3.2.2, we propose a replay grounding for the replay annotations, where the objective is to link each replay to its corresponding live action.

7.3.1 An exhaustive list of actions to spot

We manually analyzed several soccer games and noted the different game events that typically occur. In total, we retained 17 classes of game events that we believe are essential for understanding soccer games. These game events are annotated using a single timestamps with a one second precision. However, unlike the 3 classes of SoccerNet, most of the new game events are not registered on sports websites. Therefore, we need to manually annotate all of these actions. To do so, thanks to the financial support of the Research and Technologies Department of Wallonia, Belgium, and the University of Aalborg, Denmark, 20 humans annotators were hired to go through the entire video dataset, for a total of up to 800 hours of annotations. All classes and the definition of their corresponding timestamps are listed hereafter[2,3]:

1. *Penalty*: the exact moment at which the penalty is shot by the player.
2. *Kick-off*: the exact moment at which the pass in the central circle is made by a player.
3. *Goal:* the exact moment at which the ball crosses the goal line (as in SoccerNet [72]).
4. *Substitution:* the exact moment at which the entering player crosses the outer field line to enter the field and replace one of his teammates (as in SoccerNet [72]).
5. *Offside*: the exact moment at which the linesman raises his flag to signal an offside.
6. *Shots on target*: the exact moment at which the attacking player shoots the ball at the goal frame.

[2] For a rigorous definition of each action, we recommend the reader to look at the *rules of the game* provided by the International Football Association Board (IFAB) [93] as it contains the exact definition of such actions.

[3] The notion of *exact moment* corresponds to a timestamp expressed in millisecond.

7. *Shots off target*: the exact moment at which the attacking player shoots the ball at the goal frame but misses it.
8. *Clearance*: the exact moment at which the goal keeper shoots the ball, after the ball passed behind his end line.
9. *Ball out of play*: the exact moment at which the ball exits the field by the side or end line.
10. *Throw-in*: the exact moment at which the ball leaves the hands of the player.
11. *Foul*: the exact moment at which the foul is committed, only if the referee whistles.
12. *Indirect free-kick*: the exact moment at which the free-kick is shot. We consider the free-kick as indirect when there is no intention of directly scoring with a single shot. Usually, there is no wall from the defending team and a simple pass is made to a teammate.
13. *Direct free-kick*: the exact moment at which the free-kick is shot. We consider the free-kick as direct when there is an intention of directly scoring with a single shot (or a small pass directly followed by a shot). Usually, there is a wall from the defending team and a direct shot is made towards the goal.
14. *Corner*: the exact moment at which the corner is shot by the attacking player.
15. *Yellow card:* the exact moment at which the referee shows the yellow card to the player (as in SoccerNet [72]).
16. *Red card*: the exact moment at which the referee shows the red card to the player.
17. *Second yellow that leads to a red card*: the exact moment at which the referee shows the first yellow card to the player.

Alongside these timestamps, we also annotate the *visibility* or *unshown* characteristic of the game event in the TV broadcast. In fact, many unimportant actions such as ball out of play or clearances are not shown on TV and are replaced by views of the coaches, the public or a replay of a more interesting action. By annotating these unshown events as well, we are able to evaluate the methods on their capacity to spot events, only based on the context rather than direct visual cues.

As a bonus, we also annotate information related to which team performs the action, either the *home* or *away* team. We anticipate that it might be useful for future applications. Furthermore, the choice of this particular set of game events perfectly defines a split between *in-play* and *paused game*. In fact, paused game only starts with actions (3), (5), (9) and (11) and ends with actions (1), (2), (6), (7), (8), (10), (12), (13) and (14). This could lead to an additional temporal segmentation task, whose objective is to segment the game between in-play and paused.

Figure 7.1 shows some statistics related to the new annotations. We inspect the number of annotated action spots for each class and their respective proportions of visible and unshown instances. As can be seen, a large majority of actions are shown on the TV broadcast except for the kick-offs, clearances and throw-ins. This can be explained by the fact that these actions are relatively unimportant for the viewer as they convey no particular emotions and do not change the stakes of the game. On the contrary, penalties, goals, shots and direct free-kicks are almost always visible since they correspond to very important events and are appreciated by the viewers. In total, we annotated more than a 110,703 individual action spots, which is more than 16 times the amount of annotations from the original SoccerNet dataset.

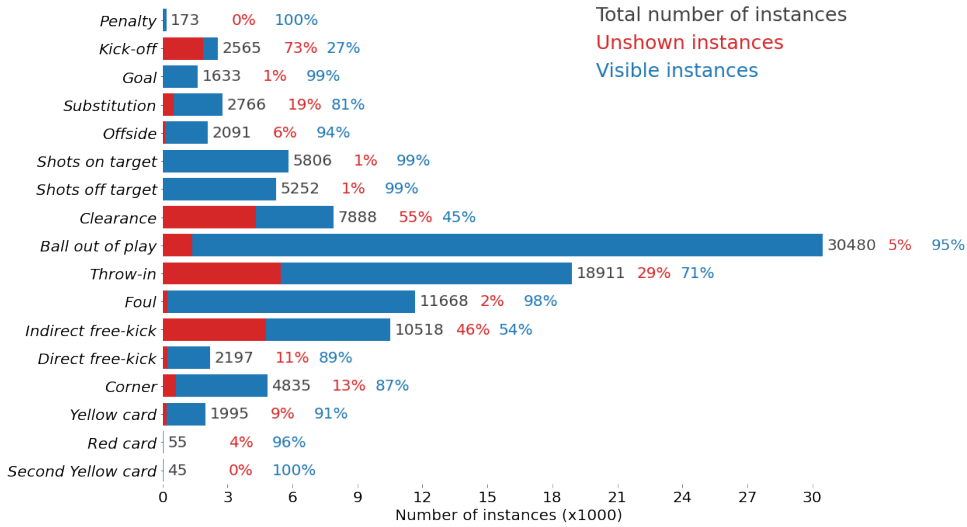


Figure 7.1: Statistics on action spots. We provide the total number of annotated instances for each class separately, their respective percentage of unshown instances and visible instances on the TV broadcast.

7.3.2 Editing semantics

7.3.2.1 Camera shot segmentation and boundary detection

We start by defining the different classes of camera shots that can be seen in regular soccer TV broadcasts. We retained 13 different classes of cameras that can be separated into three main categories: the main cameras filming the field from above with a large field of view, the close-up cameras filming the field from ground level with a narrow field of view, and some ambiance or artistic cameras for emotional content. Since annotating

this type of information was tedious and time consuming, we only annotated a subset of 200 videos from the original SoccerNet data using the same task force of students for another 800 hours of annotations. The different camera shot classes are defined hereafter and a typical image from each of these cameras is shown in Figure 7.2.

1. *Main camera center*: the camera that is shown most of the time. It is placed high in the stadium and is centered on the middle field line. It films the players with a wide field of view which allows an easy understanding of the course of the game.

2. *Main camera left*: this camera is placed high in the stadium on the left side of the field. It is mostly used to get an easy overview of what is happening close to the left goal. It can also be used sometimes to show the right side of the field from a further perspective, mostly for an artistic effect. It is also sometimes called the 16-meter left camera.

3. *Main camera right*: this camera is the counterpart of the main camera left on the right side of the field.

4. *Main behind the goal*: this camera is placed behind the goal, either on a moving crane or in the stadium. It allows to get a perpendicular field of view compared to the other main cameras.

5. *Goal line technology camera*: this camera is often placed next to the main camera left or right, but is aligned with the goal line. It is used to check if the ball entirely crossed the line in contentious goal cases.

6. *Spider camera*: this camera is placed above the field and can move freely in 3 dimensions thanks to long cables. It is often used in replays for a dynamic immersion in the action.

7. *Close-up player or field referee*: These cameras are on ground-level, either fixed or at the shoulder of an operator. They film the players or the referees on the field with a narrower field of view.

8. *Close-up side staff*: This camera's only purpose is to film the reaction of the coaches and the staff outside of the field. This also includes players on the bench or warming-up.

9. *Close-up corner*: these cameras are often on the shoulder of an operator and film the player that shoots the corner.

10. *Close-up behind the goal*: these cameras are either on the shoulder of an operator or fixed on the ground and film the goal keeper or the players from behind the goal.

11. *Inside the goal*: these cameras are placed inside of the goal and are sometimes shown during replays for an artistic effect.

12. *Public*: these cameras are placed at different places in the stadium and aim at filming the reaction of the public.

13. *Other*: all other types of cameras that may not fit in the above definitions and that are most often used for artistic effects (*e.g.*, the helicopter camera or a split screen to show simultaneously two different games).

While determining the camera class, we also annotate the boundaries between the different shots as well as the type of transition between them. We noted three types of transitions between the shots, as illustrated in Figure 7.3, that are listed hereafter:

1. Abrupt: when there is a clean cut between the two camera shots.

2. Smooth: when there is a fade out of the previous camera shot to the next.

3. Logo: when there is a logo in between the transitions.

Some statistics about classes of camera shots are shown in Figure 7.4. As can be seen, the distribution between the different cameras is highly unbalanced. As expected, the main camera center is shown most of the time, followed by the close-up player or referee camera. In total, 158,493 camera shots and their temporal boundaries were annotated. Regarding these camera changes, 55% correspond to abrupt transitions, 18% to smooth transitions and 27% to logo transitions.

Finally, to evaluate the performance of the camera shot temporal segmentation, we suggest to use the F_1 score defined in Chapter 3. The camera shot boundary detection problem can be assimilated to a spotting task, with a single class being the change of camera shot. Therefore, we suggest to use the Average Precision (AP) for a small value of δ (*e.g.*, 1 second).

7.3.2.2 Replay grounding task

As a final type of high-level semantics, we also annotate whether the camera shot corresponds to a *replay* of a previous action or if it corresponds to the *live* stream. This information is annotated for all videos of the original dataset as unlike camera shot classes and boundaries, it is much faster to annotate. In the case where the camera shot is a replay of an annotated action, we annotate a link between this replay and its corresponding action. This information is valuable in the case of automatic production or highlights generation as being able to detect the replay of a specific action can be a sign of importance of this action in a game. Thanks to these new annotations, we propose a new replay grounding task defined as follows:

Figure 7.2: Illustrations of the camera shot classes. We showcase one example per camera class of a typical view from this camera. The numbers correspond to the classes defined in the section. Sources: CANAL+ and EUROSPORT.

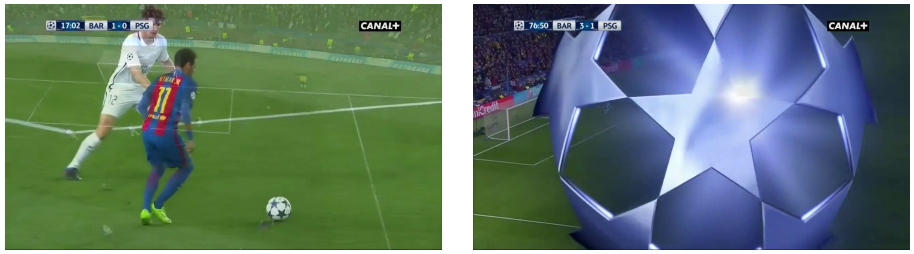


Figure 7.3: Illustrations of the shot transitions. We showcase one example for the smooth transition (left) and one for the logo transition (right). Source: CANAL+.

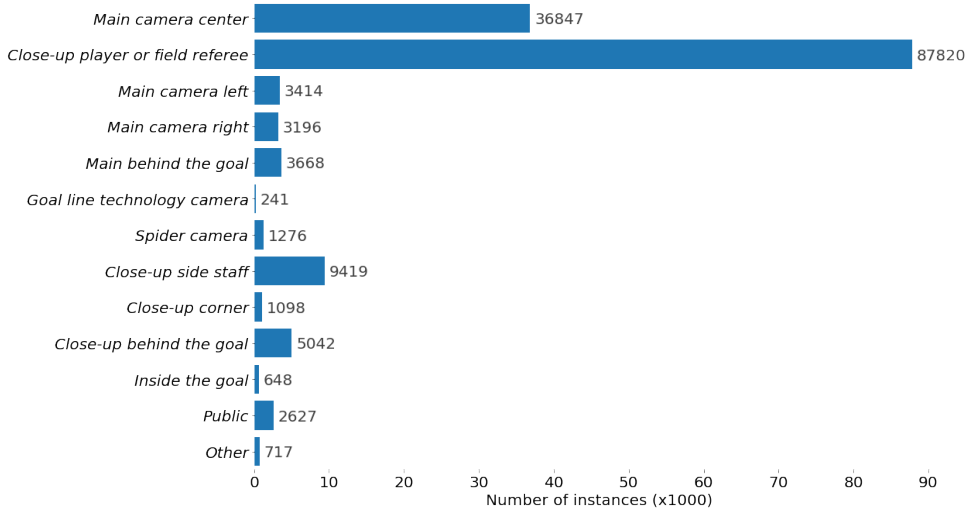


Figure 7.4: Statistics on camera shots. Per-class statistics on the total number of annotated instances of each class.

REPLAY GROUNDING TASK

Given a camera shot containing a replay of a live action, spot the moment this action occurred live in the video.

More than 32,000 camera shots corresponding to replays linked to an action spot were annotated. The distribution between the categories of action and the types of camera shot is shown in Figure 7.5. As expected, the most commonly shown actions in replays are the goals and the fouls. We can also see that the distribution of the camera shot classes showing the replays is very different than the one prior on these classes shown in Figure 7.4. Cameras such as the main camera left and right or the close-up behind the goal are much more used in replays than in live sequences.

Finally, to evaluate the performances on this task, we suggest to use the same evaluation metric as the one for action spotting, with a single spotting class (average-AP). In fact, this the task can be viewed as a conditional action spotting task, whose objective is to spot one action per replay in each game.

To summarize this section, the different tasks are illustrated in Figure 7.6. We believe that there exist a correlation between these tasks so that multi-task approaches could benefit from the different types of annotations.

7.4 Experiments on the use of CALFNet for action spotting

In the following, we provide some results for the action spotting task that show that our method presented in Chapter 6 achieves good results for 17 types of actions as well.

Since CALFNet was originally designed for action spotting, only the value of the *number of classes* parameter needs to be adapted (see Appendix A.2 for details). Also, the K parameters have to be chosen for each class. To do so, we perform a Bayesian optimization as explained in Section 6.4.1.2. Once the hyper-parameters are fixed, we train several networks and average their performance on the test set. After doing so, our method achieves an average-mAP of 41.6%. We also provide the average-AP per class in Figure 7.7. As can be seen, classes such as corner and goal perform well while red and second yellow cards are very hard to retrieve. This discrepancy can be explained by two main factors. First, certain classes of game events have clear visual cues, which is for example the case of corners and free-kicks. Second, as we have seen in Section 7.3.1, there is a high imbalance between the classes. In fact, classes with fewer samples tend to perform worse than the ones with higher number of samples.

Also, we evaluate our spotting results separately on the visible and unshown action spots. To do so, we first match each prediction of the network with its closest ground-truth action spot and then compute the average-mAP separately for those associated with visible actions and with unshown actions. For the unshown actions, since there are

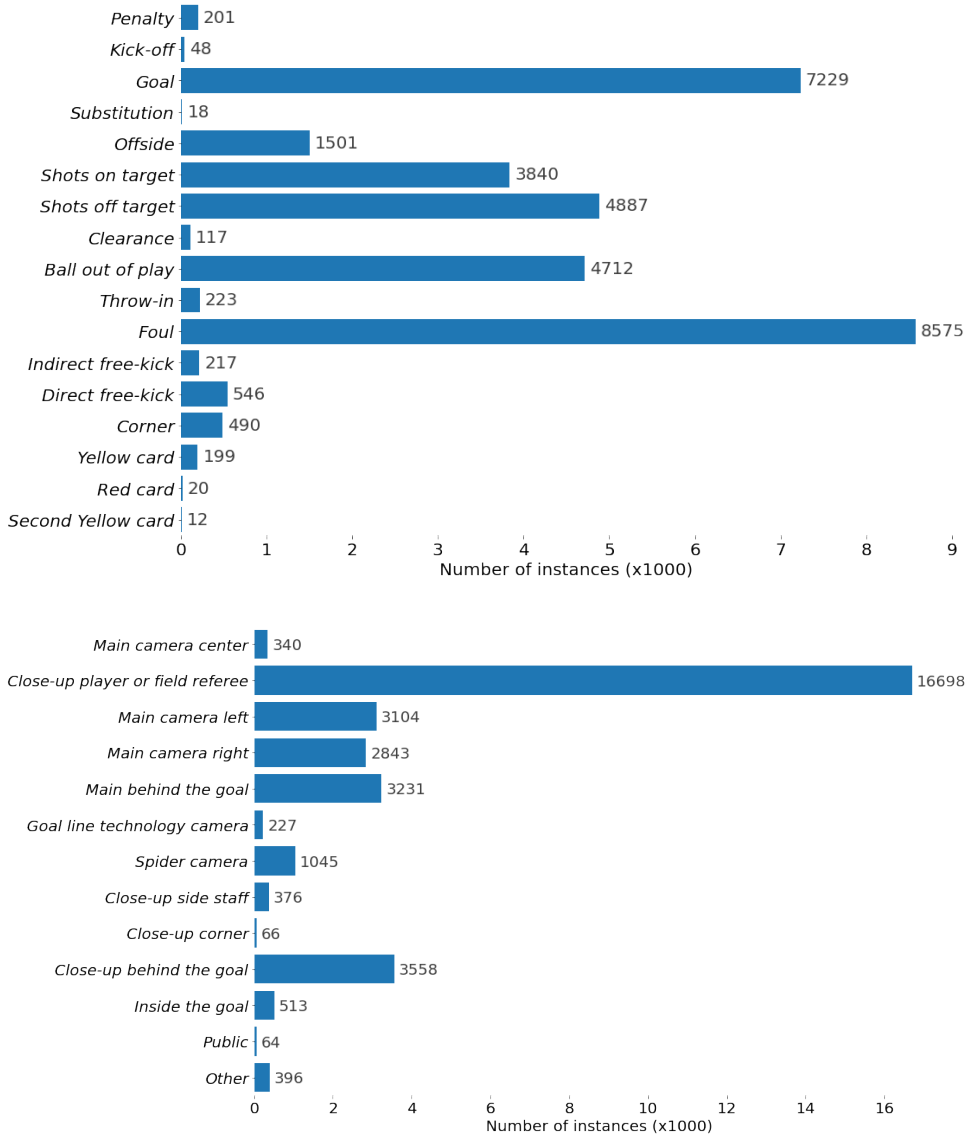


Figure 7.5: Statistics on replays. Statistics on the total number of annotated instances of replay for each action class (top) and each camera shot class (bottom).

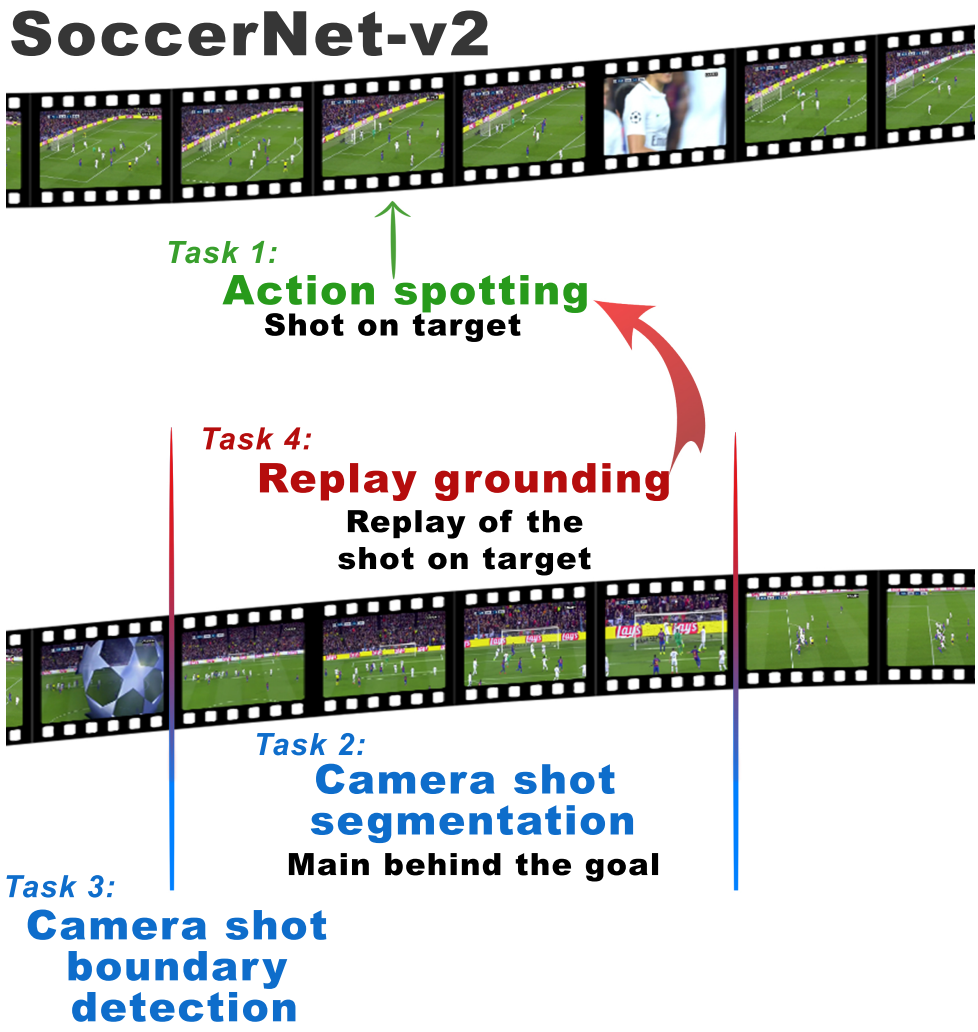


Figure 7.6: Summary of new tasks. Our proposals enable the training of 4 high-level semantic tasks on the coming SoccerNet-v2 dataset: action spotting, to detect all game events in the soccer game defined by single timestamps, camera shot segmentation to assign a label to each frame corresponding to the type of camera, camera shot boundary detection to spot all camera transitions, and replay grounding to link each replay to its corresponding live action.

no ground-truth instances of penalties, goals, red and second yellow cards, we remove these classes from the evaluation. Following this evaluation methodology, we achieve 43.95% of average-mAP for the visible action spots and 37.8% for the unshown ones. This shows that even though visible actions are better retrieved, our context-aware network is still able to correctly find some of the unshown action spots since it uses the temporal context surrounding the actions.

Finally, we show some qualitative results in Figure 7.8 for each class on one game of the test set. As can be seen, all corners of that game are retrieved, while shots on and off target are most of the time not correctly retrieved. This shows that there is still room for improvement.

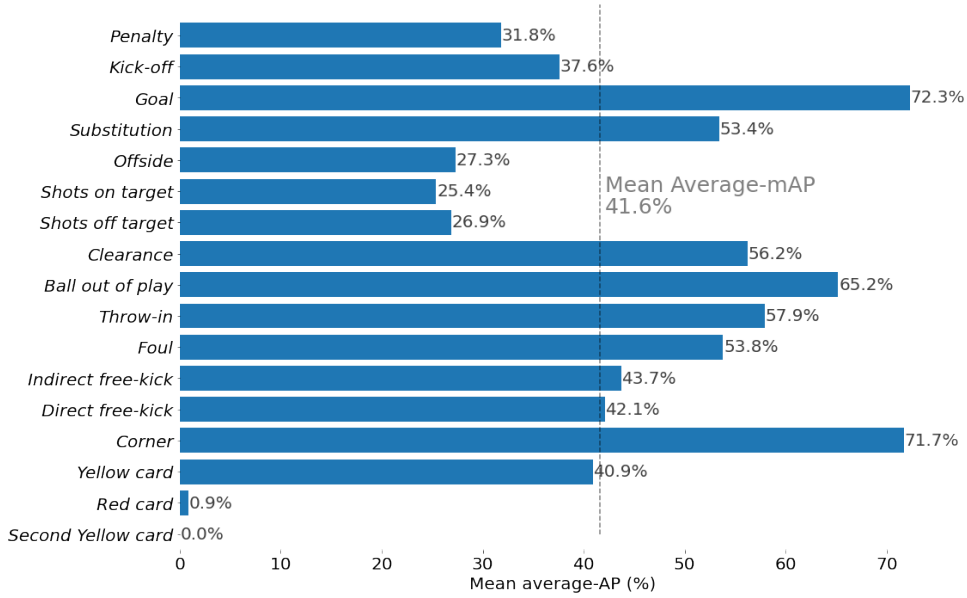


Figure 7.7: Performance per class. We compare the average-AP for each of the 17 action classes. Some classes seem easier to spot than others, which can be explained by more discriminant visual cues and the distribution of classes in the dataset.

7.5 Conclusion and future works

In this chapter, we have described a proposal for the extension of the SoccerNet dataset, comprising an exhaustive set of game events along with novel annotations and tasks related to the TV broadcast. In particular, we increase the number of annotations for the action spotting task by more than 16 times and proposed extra annotations related to the

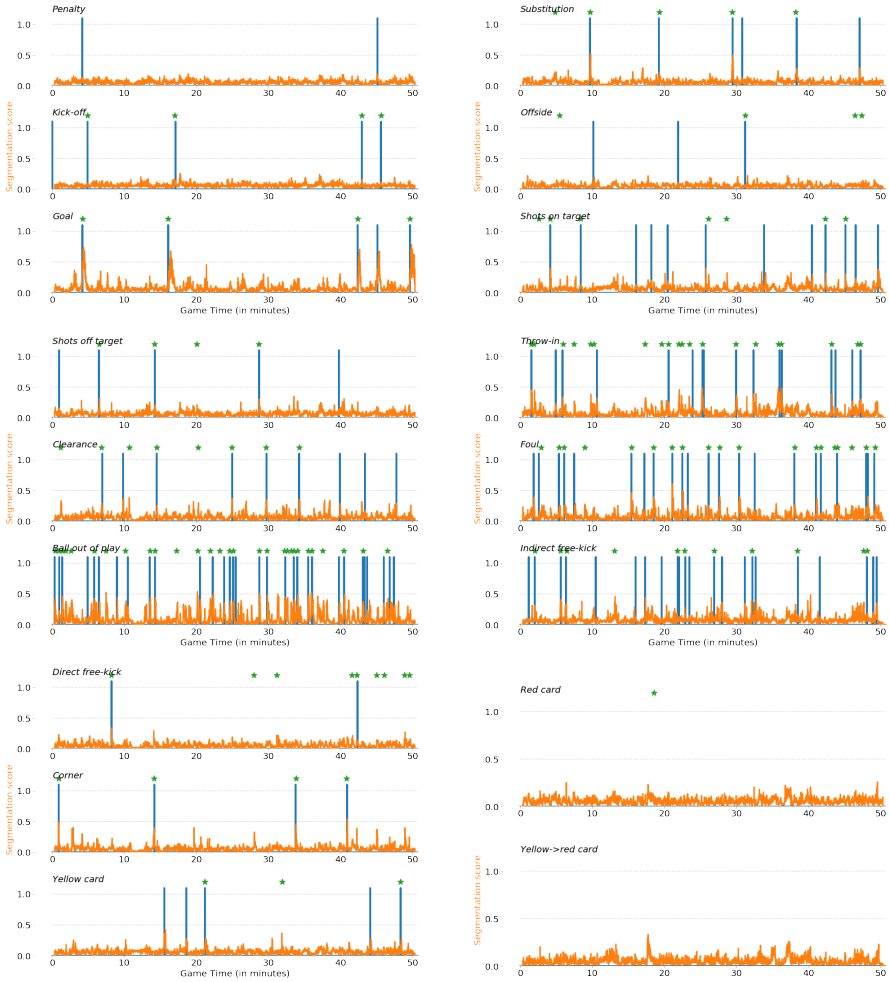


Figure 7.8: Qualitative results. **Temporal segmentation curves** and **spotting results** for each class on one half of a game. The **Ground-truth actions** are also shown.

TV broadcast. We also define four tasks on this dataset, including the novel task of replay grounding.

Regarding real-world applications, the possibilities to leverage the annotations of this dataset are numerous. As a first future work, we suggest to investigate the automatic highlights generation task. In fact, by cascading the different stages of the presented tasks, one could first gather all action spots occurring during the game, then correctly cut the clips using a boundary detection method and use the temporal segmentation to classify these segments between the different cameras. This is a valuable information as it is often recommended to first show the action from a main camera perspective so that the viewer can easily understand what is going on. Finally, replays about a specific action indicate which actions are the most important during the game; this knowledge is essential when choosing which actions to show in a selection of highlights.

As a second possible work, we expect that the task of camera shot prediction could be useful for automatic game production. In fact, training a method to predict when to switch between cameras, and to which type of camera, is an interesting topic for further research. Furthermore, we could also investigate the task of predicting when a replay of a particular action should be shown.

CHAPTER 8

Conclusion

Throughout this thesis, we have explored the notion of semantics in video sequence from low level, describing the natural content of the scene, to high level, interpreting the scene. In particular, our innovative real-time methods solve multiple semantic-related tasks, which constitute valuable assets for video applications.

In the first part, we investigate the extraction of low-level semantics. In Chapter 2, we present an asynchronous combination method for the task of background subtraction which provides real-time motion detection maps using the information provided by a slow, but effective, semantic segmentation network. Our method achieves performances close to the state of the art while being real time, which is not the case for the current best algorithms. For future investigations, one could generalize the asynchronous combination table for other tasks as well, such as combining a fast background subtraction algorithm with a slow optical flow network for motion detection or a fast optical flow network with a slow semantic segmentation network for optical flow computation. In Chapter 3, our online knowledge distillation method, in which a fast student network is continuously trained using surrogate ground-truth annotations provided by a slow, but performant, teacher network, introduces a new paradigm for training networks. We showcase this method for the task of semantic segmentation of the player on the field, and demonstrate that a real-time network reaches performances close to its non-real-time teacher and even sometimes surpasses it. As an extension, we showcase a second use case for online distillation in Chapter 4, in which the student and teacher networks share different modalities and fields of view of the same scene. Particularly, our novel custom data augmentation strategy helps the student network to detect soccer players on the whole fisheye image even though surrogate ground truths were only available in a restricted zone of its field of view.

In the second part, we focus on high-level semantics. We start with a bottom-up approach that leverages low-level semantics to extract higher levels of semantics in Chapter 5. In particular, the semantic segmentation masks of the players, the field and the field lines are used to derive the direction and group of the players as well as some information on the view of the camera. From these, we engineer a semantic-based decision tree to segment four major game phases: goal or goal opportunity, attack, middle and defense. Then, in Chapter 6, we spot game events based on the temporal context directly captured inside of a neural network. This is achieved through our custom loss function that evolves smoothly around the ground truth action spots. Thanks to these new developments, our method achieves state-of-the-art performances on the SoccerNet dataset,

beating every other methods by far. As a final work, we expend the range of high-level semantics in SoccerNet by proposing novel semantic annotations. We define a complete set of action for the action spotting task, comprising the 17 most common game events in soccer. Alongside, new annotations on semantics linked to the editing of the TV broadcast are provided, by proposing camera shot boundaries and classes among a set of 13 different camera types. Furthermore, the camera shots corresponding to replays are also annotated, which allows us to define a novel replay grounding task aiming to link each replay to its corresponding action.

Overall, we have proposed efficient methods to extract different levels of semantics and shown that combining them improves the quality of the extracted semantics to the level of real-time state-of-the-art performances. While there is still room for improvements and innovations, our methods are effective tools for new industrial opportunities. Also, our proposals for extending the SoccerNet dataset are steps towards a better automatic interpretation of soccer games. For example, the new types of high-level semantics enable to work towards the task of automatic highlights generation. Finally, as an interesting research topic, one could generalize the concept of online distillation to all sorts of tasks, especially for the many situations where real-time performance are required or when ground-truth annotations are not available. It is our hope that the work presented in this thesis will contribute to a better understanding of video of soccer games and, why not, video understanding in general.

Part III
Appendices

APPENDIX A

Description of the networks

Contents

This appendix provides some details about the implementations of the networks that are presented in this document. More specifically, it provides a complete description of our own real-time semantic segmentation network called TinyNet used in Chapter 3 and Chapter 5, and our action spotting network CALFNet 6 used in Chapter 6 and Chapter 7.

A.1 Real-time semantic segmentation network: TinyNet

First, we present our own lightweight deep learning semantic segmentation network, called TinyNet. Let us recall that semantic segmentation is an image segmentation technique that aims at labeling each pixel according to a set of classes. Many works are carried out in that field. Region proposal-based techniques such as DeepMask, SharpMask and Region-CNN developed by Facebook AI Research [147] are common approaches in semantic segmentation. More recently, deep learning networks have been developed such as PSPNet [221] or Mask R-CNN [78]. These networks automatically label each pixel according to hundreds of different classes, but since they are meant to be universal, they also are very slow to segment images.

TinyNet is a network architecture whose design follows the same principles as the universal network PSPNet which is mainly composed of a pre-trained ResNet model that extracts the feature maps, a pyramidal pooling module to collect global context information [221], and an upsampling procedure. We designed this lightweight adaptation to ensure real-time inference for the scene-specific case of soccer.

ResNet was introduced by He *et al.* [79]. The idea is that, rather than directly approximating the input-output function as in conventional networks, the network approximate the residual function that has to be added to the input to give the correct output. To do so, ResNet modules have two branches, one performing non-linear operations to approximate the residual function and one that bypasses all the connections from the input. The

underlying equation is then defined as

$$H(x) = F(x) + x, \tag{A.1}$$

where $H(x)$ is the original function to approximate and $F(x)$ is the residue that has been added to the input. An illustration of a typical ResNet module is shown in Figure A.1. ResNet modules have the advantage of alleviating the problem of increased optimization difficulty when adding new layers in a network.

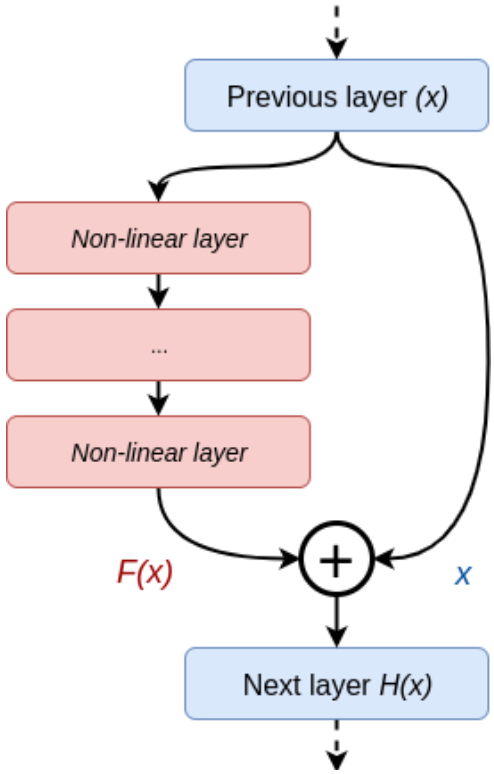


Figure A.1: Typical ResNet module. The function $H(x)$ is approximated using the input x and a residual function $F(x)$.

The goal of the pyramidal pooling module introduced with PSPNet is to segment the image in regions of different sizes in order to retrieve context information. The module is composed of several branches which reduce the feature maps obtained by the ResNet module into regions by average pooling. In the original PSPNet network, the image is pooled into 1×1, 2×2, 3×3 and 6×6 regions, which are designed for square images of relatively low resolution, in order to get global information on what the scene represents.

The network that we developed for this work is a scaled-down version of the PSPNet network. In fact, PSPNet is meant to be used for universal segmentation, implying that it must be robust for a large variety of scenes. In the particular case of soccer, we only have a few classes compared to the hundreds of classes that PSPNet is capable to deal with. Also, the background and the objects of interest are roughly the same from one game to another. By training the model only on images of soccer, that is by being *scene-specific*, we can further increase the performances of the network by discarding elements it will never see. This has the advantage of increasing the performances for our particular case while decreasing the complexity and the computation time of the network. To do so, we design one small ResNet module that we scale down in terms of the number of parameters and remove some of the first convolutional layers. The pyramidal module is also adapted for the main camera of a soccer game. In our case, we have full-HD wide angle images rather than 473×473, meaning that the pooling module has to divide the image a bit more horizontally than vertically. We modify the pyramid pooling module to have 1×1, 3×2, 5×3 and 9×4 regions as shown on Figure A.2, which illustrates the entire network. By doing so, we managed to divide the number of parameters by a factor 100 compared to the original PSPNet which results in an drastic increase in training and inference speed. The architecture of the network is depicted in Figure A.3.

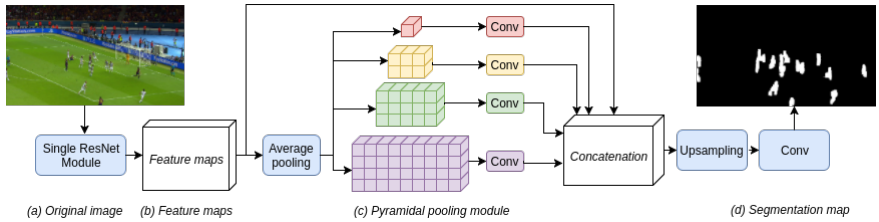


Figure A.2: Overview of our semantic segmentation network architecture. We have four components: (a) the original image that will be segmented into lines or players, (b) a single ResNet module with convolution layers in order to extract the feature maps, (c) the pyramidal module to gather context information, and (d) the upsampling procedure that produces the final segmentation map, here showing the player segmentation. The network is inspired by the design of the PSPNet network [221].

A.2 Action spotting network: CALFNet

The architecture of the network used in Chapter 6 and Chapter 7 for the action spotting task on SoccerNet is depicted in Figure 6.3 and Figure A.4. We use the following notations for the layers of a convolutional neural network:

- FC(n) is a fully connected layer (*e.g.*, in a multi-layer perceptron) between any vec-

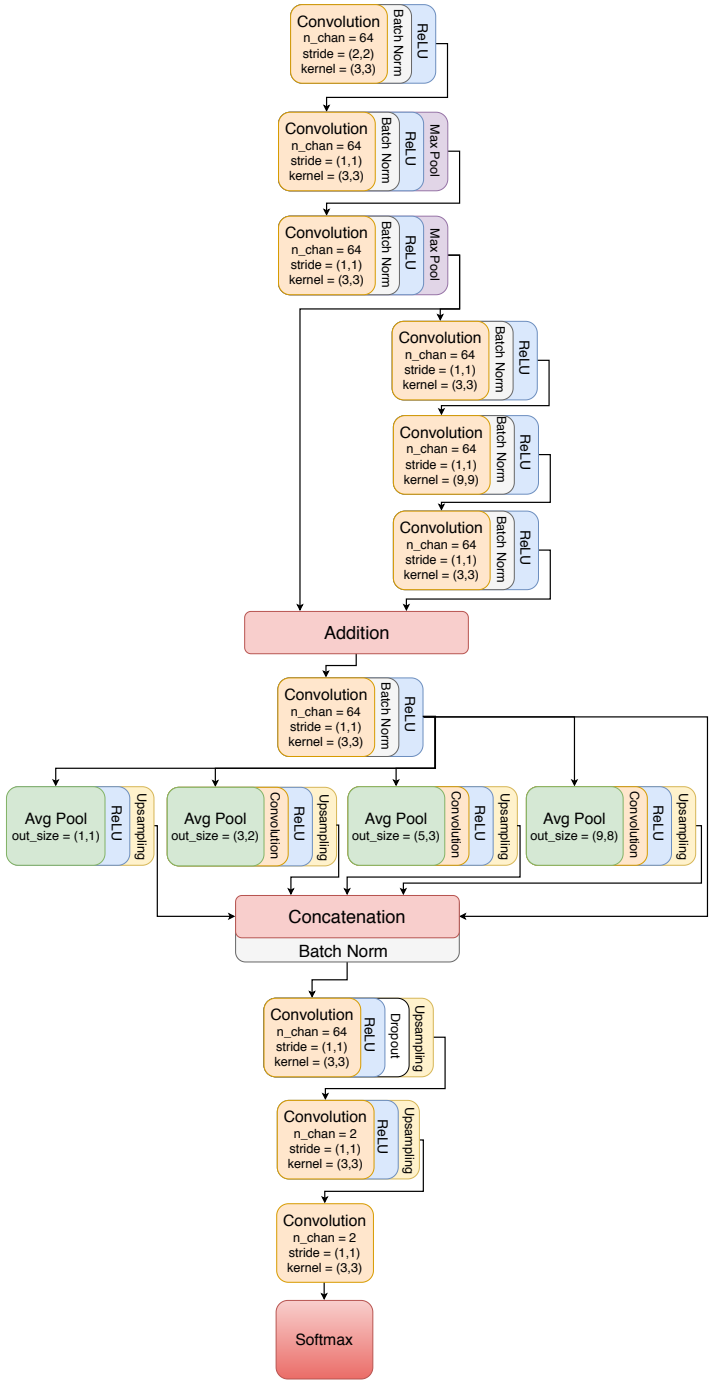


Figure A.3: Architecture of our real-time segmentation network TinyNet. It is a lightweight version of PSPNet 101 with 100 times less parameters.

tor to a vector of size n.

- ReLU is the rectified linear unit.
- Conv$(n, p \times q)$ is a convolutional layer with n kernels of dimensions $p \times q$.
- C is the number of classes in the spotting task.
- N_I is the number of features per frame.
- N_F is the number of frames in the considered chunk.
- N_{GT} is the number of ground-truth actions in the considered chunk.
- N_{pred} is the number of predictions output by the network for the spotting task.
- f is the number of features computed for each class, for each frame, before the segmentation module.
- r is the temporal receptive field of the network (used in the temporal convolutions).
- $\hat{\mathbf{Y}}$ regroups the spotting predictions of the network, and has dimension $N_{\text{pred}} \times (2 + C)$. The first column represents the confidence scores for the spots, the second contains the predicted locations, and the other are per-class classification scores.
- $\mathbf{Y}$ encodes the ground-truth action vectors of the chunk considered, and has dimension $N_{\text{GT}} \times (2 + C)$.
- K_i^c $(i = 1, 2, 3, 4)$ denotes the context slicing parameters of class c.

1. Frame feature extractor and temporal CNN. SoccerNet provides three frame feature extractors with different backbone architectures (I3D, C3D, and ResNet). Each of them respectively extracts 1024, 4096, and 2048 features that are further reduced to 512 features with a Principal Component Analysis (PCA). We use the PCA-reduced features provided with the dataset as input of our temporal CNN.

The aim of the temporal CNN is to provide Cf features for each frame, while mixing temporal information across the frames. It transforms an input of shape $N_F \times 512$ into an output of shape $N_F \times Cf$.

First, each frame is input to a 2-layer MLP to reduce the dimensionality of the feature vectors of each frame. We design its architecture as: FC(128) - ReLU - FC(32) - ReLU. We thus obtain a set of $N_F \times 32$ features, which we denote by $\mathcal{F}_{\text{MLP}}$.

Then, $\mathcal{F}_{\text{MLP}}$ is input to a spatio-temporal pyramid, *i.e.* it is input in parallel to each of the following layers of the pyramid:

- Conv$(8, r/7 \times 32)$ - ReLU

- Conv(16, $r/3 \times 32$) - ReLU
- Conv(32, $r/2 \times 32$) - ReLU
- Conv(64, $r \times 32$) - ReLU

producing $8+16+32+64 = 120$ features for each frame, which are concatenated with $\mathcal{F}_{\text{MLP}}$ to obtain a set of $N_F \times 152$ features.

Finally, we feed these features to a Conv($Cf, 3 \times 152$) layer, which produces a set of $N_F \times Cf$ features, noted $\mathcal{F}_{\text{TCNN}}$.

2. Segmentation module. This module produces a segmentation score per class for each frame. It transforms $\mathcal{F}_{\text{TCNN}}$ into an output of dimension $N_F \times C$, through the following steps:

- Reshape $\mathcal{F}_{\text{TCNN}}$ to have dimension $N_F \times C \times f$.
- Use a frame-wise Batch Normalization.
- Activate with a sigmoid so that each frame has, for each class, a feature vector $v \in (0,1)^f$.
- For each frame, for each class, compute the distance d between v and the center of the unit hypercube $(0,1)^f$, *i.e.* a vector composed of 0.5 for its f components. Hence, $d \in [0, \sqrt{f}/2]$.
- The segmentation score is obtained as $1 - 2d/\sqrt{f}$, which belongs to $[0,1]$. By doing so, scores close to 1 for a class (*i.e.* v close to the center of the cube) can be interpreted as indicating that the frame is likely to belong to that class.

The segmentation scores ζ_{seg} output by the segmentation module thus have dimension $N_F \times C$ and are assessed through the segmentation loss $\mathcal{L}^{\text{seg}}$.

3. Spotting module. The spotting module takes as input $\mathcal{F}_{\text{TCNN}}$ and ζ_{seg}, and outputs the spotting predictions $\hat{\mathbf{Y}}$ of the network. It is composed of the following layers:

- ReLU on $\mathcal{F}_{\text{TCNN}}$, then concatenate with ζ_{seg}. This results in $N_F \times (Cf + C)$ features.
- Temporal max-pooling 3×1 with a 2×1 stride.
- Conv($32, 3 \times (Cf + C)$) - ReLU
- Temporal max-pooling 3×1 with a 2×1 stride.
- Conv($16, 3 \times 32$) - ReLU
- Temporal max-pooling 3×1 with a 2×1 stride.

- Flatten the resulting features, which yields $\mathcal{F}_{\text{spot}}$.
- Feed $\mathcal{F}_{\text{spot}}$ to a FC($2N_{\text{pred}}$) layer, then reshape to $N_{\text{pred}} \times 2$ and use sigmoid activation. This produces the confidence scores and the predicted locations for the action spots.
- Feed $\mathcal{F}_{\text{spot}}$ to a FC(CN_{pred}) layer, then reshape to $N_{\text{pred}} \times C$ and use softmax activation on each row. This produces the per-class predictions for the action spots.
- Concatenate the confidence scores, predicted locations, and per-class predictions to produce the spotting predictions $\hat{\mathbf{Y}}$ of shape $N_{\text{pred}} \times (2 + C)$.

Eventually, $\hat{\mathbf{Y}}$ is assessed through the action spotting loss $\mathcal{L}^{\text{as}}$.

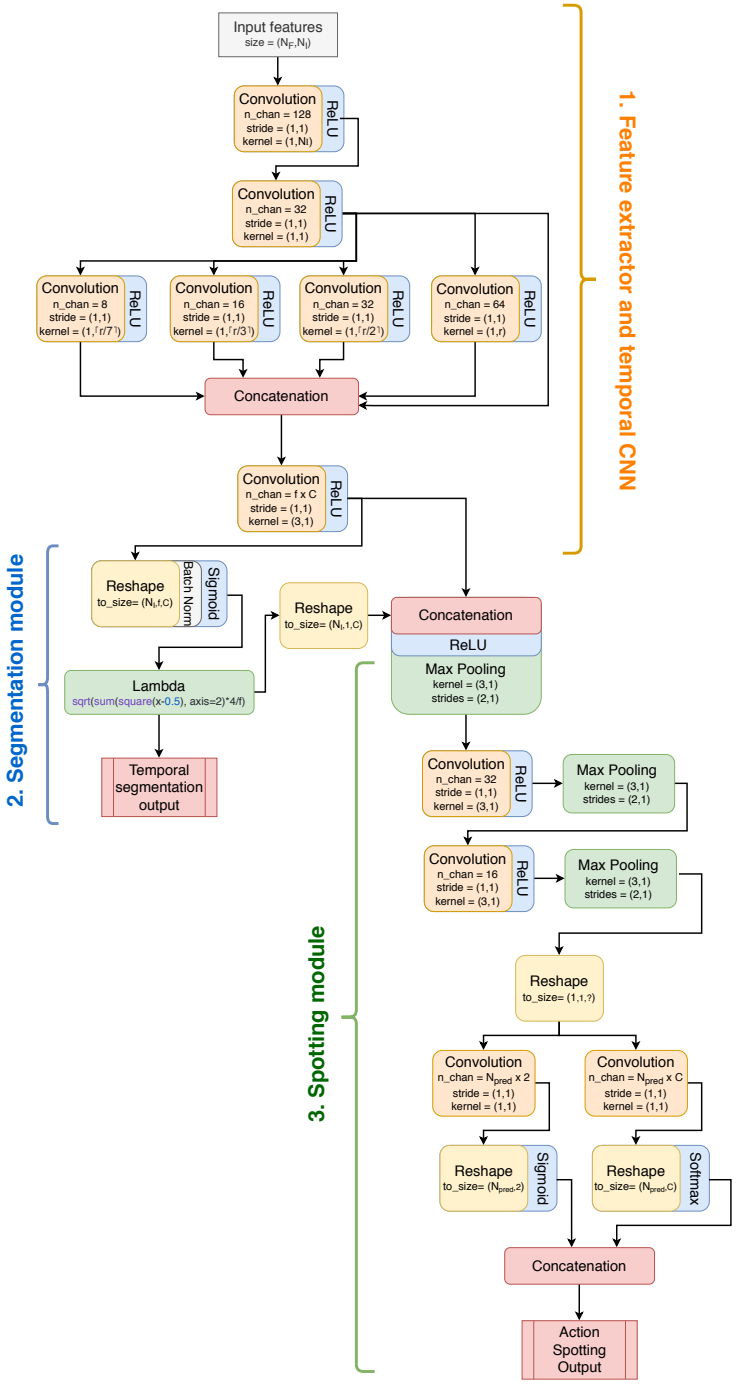


Figure A.4: Architecture of our action spotting network called CALFNet. It is composed of both a segmentation and action spotting module.

APPENDIX B

Online distillation: additional details and experiments

Contents

This appendix provides further details on the method presented in Chapter 3. Particularly, Section B.1 describe the dataset used for training $\mathcal{S}_{\text{pretrained}}$ with offline distillation and we provide some additional experiments and results in Section B.2.

B.1 Description of the dataset for the offline distillation

In this section, we provide further details about the datasets used for training, validating and testing the offline distillation process that produces the networks $\mathcal{S}_{\text{pretrained}}$ of Section 3.3. We use the main camera stream for both the soccer and basketball videos. This camera has a wide angle of view and is shown most of the time on television because it usually provides an excellent overview of the ongoing match. Hence, this camera is often used for sports analysis. Figure 3.5 shows four examples of images taken from this camera.

Regarding the soccer dataset, we use the following eight matches from the UEFA Euro 2016 competition: 1. Germany vs Slovakia; 2. Belgium vs Wales; 3. Croatia vs Portugal; 4. France vs Ireland; 5. Northern Ireland vs Wales; 6. Poland vs Portugal; 7. Switzerland vs Poland; 8. Hungary vs Belgium. We also use one match from the 2013 Belgian Jupiler Pro League, FC Bruges vs Anderlecht, in order to test the networks on a match from a different competition, for the reasons detailed in Chapter 3.

Regarding the basketball dataset, we use the following eight matches from the 2019 LNB Jeep Elite competition: 1. Bourg-en-bresse vs Cholet; 2. Le Portel vs Monaco; 3. Lyon-Villeurbanne vs Cholet; 4. Le Mans vs Châlons-Reims; 5. Gravelines-Dunkerque vs Le Portel; 6. Landerneau vs Montpellier; 7. Dijon vs Strasbourg; 8. Cholet vs Boulazac.

These two video datasets are used for training two separate instances of $\mathcal{S}_{\text{pretrained}}$, one for each sport, by usual offline knowledge distillation. We collect a set $\mathcal{X}$ of images by selecting one frame every four seconds in each video and compute their corresponding approximated ground truth $\mathcal{T}(\mathcal{X})$ using the teacher network $\mathcal{T}$. We split the dataset into three sets: a training set $\mathcal{D}_{\text{train}} = (\mathcal{X}_{\text{train}}, \mathcal{T}(\mathcal{X}_{\text{train}}))$ containing the images subsampled from the first six matches, a validation set $\mathcal{D}_{\text{val}} = (\mathcal{X}_{\text{val}}, \mathcal{T}(\mathcal{X}_{\text{val}}))$ containing the images of the seventh match and a test set $\mathcal{D}_{\text{test}} = (\mathcal{X}_{\text{test}}, \mathcal{T}(\mathcal{X}_{\text{test}}))$ containing the images of the eighth match. $\mathcal{S}_{\text{pretrained}}$ is trained on $\mathcal{D}_{\text{train}}$ using the Adam optimizer with a batch size of 1, the weighted cross entropy loss and a learning rate of 10^{-4}. We stop its training when its performances on $\mathcal{D}_{\text{val}}$, computed after each epoch, start decreasing. The good performances of $\mathcal{S}_{\text{pretrained}}$ on an unseen game, assessed on $\mathcal{D}_{\text{test}}$, confirmed that $\mathcal{S}_{\text{pretrained}}$ could be used as such for the experiments reported in Chapter 3.

B.2 Additional experiments

B.2.1 Performances with another student network

In order to demonstrate that the use of ARTHuS does not depend on our particular choice of student network, *i.e.* TinyNet, we carry out similar experiments with another student network. For that purpose, we choose ICNet from [219]. This well-known network, state-of-the-art among real-time networks on the Cityscapes dataset [39], is used for a comparison with TinyNet in this section. We adapted the implementation from the following GitHub repository: hellochick/ICNet-tensorflow from TensorFlow to PyTorch. By default, this network outputs predictions for a large variety of classes. Since we only have two classes of interest in this work, humans or background, we modified the last layer of the network so that it outputs two numbers for each pixel, after which a softmax is applied, as in TinyNet. ICNet has 6.7 million parameters, hence about 10 times more than TinyNet and 10 times less than PSPNet. We re-design the last layer of ICNet so that it considers only two classes, human or not human. On our hardware, its inference time is about 0.033 seconds per image ($\approx$ 30 fps) and its training time is 0.12 seconds per image.

The performances of ICNet as student network are compared with TinyNet in Figure B.1 for the soccer and basketball test matches. On the one hand, when ICNet is trained online from scratch, its performances are inferior to those of TinyNet also trained online from scratch. The performance curves of ICNet increase slower than those of TinyNet, which suggests that TinyNet adapts faster to the play conditions of the ongoing match, presumably because of its reduced training time. On the other hand, when

ICNet is pre-trained offline through usual knowledge distillation on the same six matches as TinyNet and then retrained online, its performances are comparable to those obtained for the same experiment with TinyNet on the soccer match and are slightly higher on the basketball match, possibly because of the higher capacity of ICNet.

Consequently, ARTHuS can be used with other student architectures such as ICNet, in which case satisfying results are also obtained. Our experiments suggest that TinyNet adapts faster and better than ICNet when trained online from scratch, while ICNet shows equivalent or better performances than TinyNet when they are retrained online from a pre-trained network. However, the inference time of TinyNet is about half ICNet's, which implies that TinyNet leaves a more comfortable amount of time for potential extra real-time analyses.

B.2.2 Analysis of a "failure" case

It can be seen in Figures 3.4 and B.1 that a drop in F_1 score occurs around $15-20$ minutes in the soccer test match and 20-25 minutes in the basketball match. We further comment this "failure" case in this section, as one might interpret it as a possible limitation of ARTHuS.

By looking at the soccer video at that moment, it can be seen that an unusual scene occurs, which shows a remote part of the field where bear-looking mascots walk and photographs, staff members and spectators sit, as depicted in Figure B.2. Fortunately, it appears that $\mathcal{S}^{\text{seg}}$ still perfectly segments the player and the referees, which are the only humans that are of interest in this scene. Hence, the drop in performances can be explained by the large number of false positives and false negatives related to the mascots and the other people external to the game on the bottom and right side of the frame, which we added in the corrected masks (and for which $\mathcal{T}$ is also confused). If we evaluate the F_1 score without taking them into account, it goes back up to the same levels of performance as those achieved for the rest of the match. For the basketball video, the scene is a time-out, in which the players are grouped together and discuss strategies, as can be seen in Figure B.2, which is a difficult case to handle for Mask R-CNN (similarly to Figure 3.6 of Chapter 3).

Therefore, we can say that the drops in performances are due to unusual scenes coupled with the evaluation method itself, rather than an actual struggle of $\mathcal{S}^{\text{seg}}$ to segment the interesting humans of the scene.

B.2.3 ARTHuS when the pre-trained model already generalizes well

As mentioned, $\mathcal{S}_{\text{pretrained}}$ is trained on six matches from the Euro 2016 competition in the case of soccer. It is interesting to check if this network generalizes well to another game of the same competition and to examine the effect of training it online with ARTHuS. To

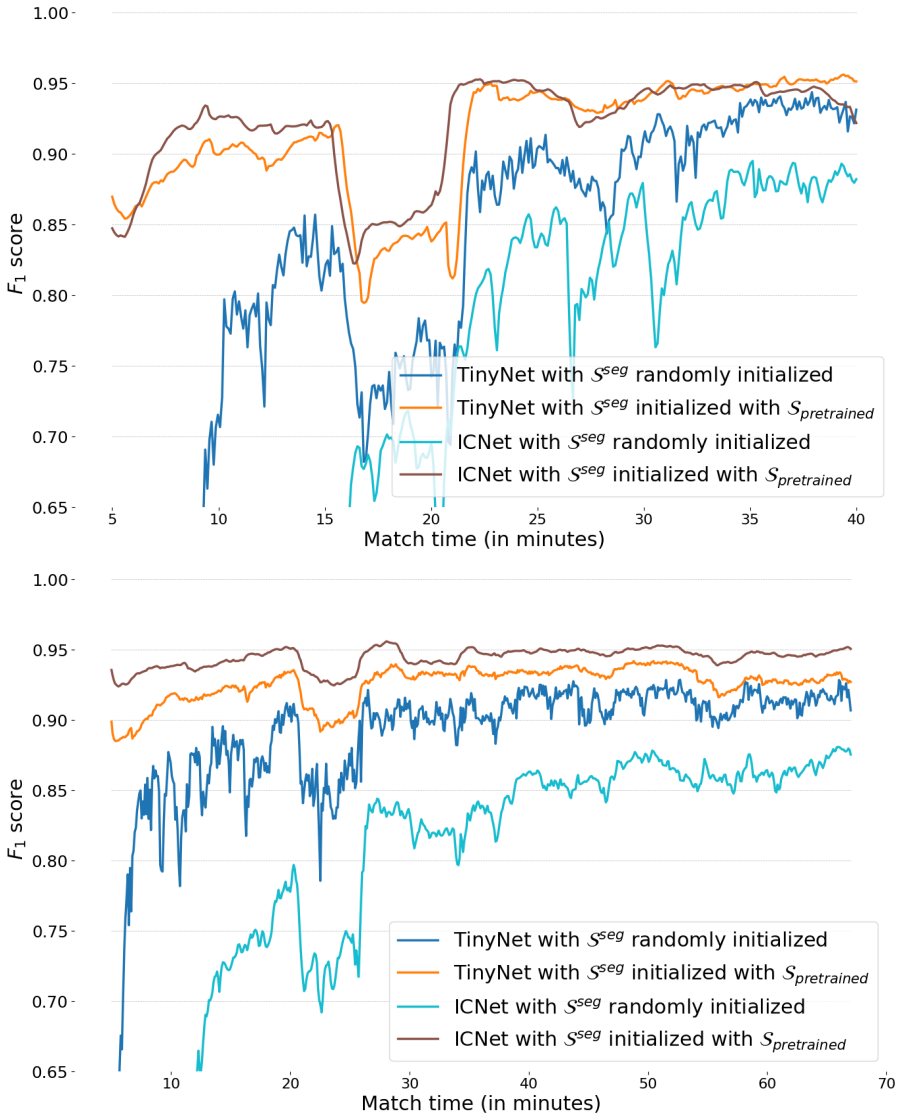


Figure B.1: Performances with another student architecture. Comparison of the performance curves obtained with different student architectures: TinyNet (*i.e.* the curves displayed previously) and ICNet [219], for the soccer (top) and basketball (bottom) test matches. $\mathcal{S}_{\text{pretrained}}$ refers to a pretrained version of the corresponding architecture, on the same set of six matches.

Figure B.2: Failure cases. Frame taken around the 20-th minute of the soccer test match (top) and the 24-th minute of the basketball test match (bottom), used to analyze the slight drops in the F_1 score at these moments. They can be explained by the unusual nature of the scene, which involves mascots and spectators close to the field in the case of soccer and a time-out in the case of basketball, and the evaluation method itself, since the humans of interest in this scene, *i.e.* the player and the referees, are still correctly segmented by $\mathcal{S}^{\text{seg}}$.

do so, we evaluate the online training with TinyNet and report the performances on the eighth match of the soccer dataset in Figure B.3. As can be seen, all curves reach about the same performances with a slight advantage for those produced by ARTHuS after 15 minutes. This indicates that, even when the network generalizes well, there is an interest in retraining it online since the performances can only increase during the match.

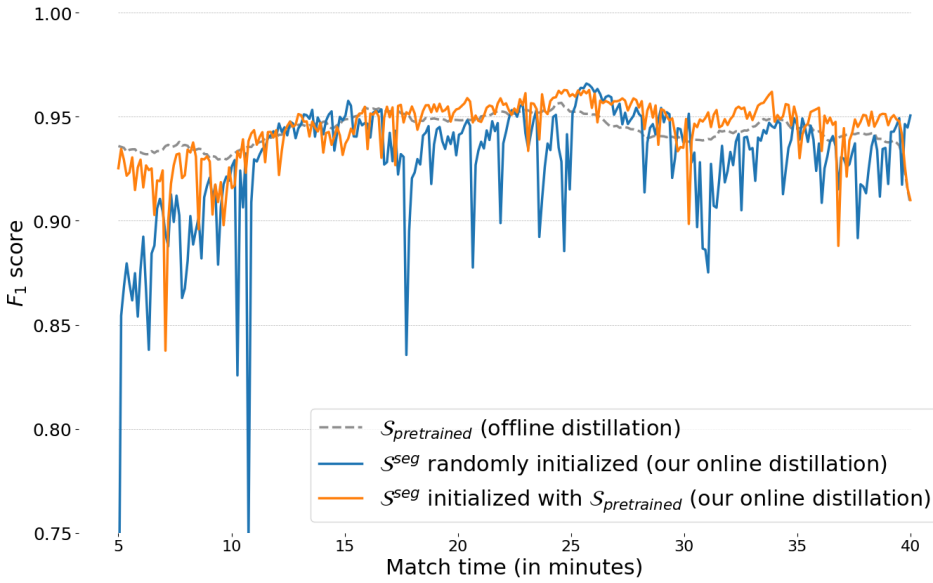


Figure B.3: ARTHuS for already good performing networks. Evolution of the performances of several variants of distilled models through their F_1 score, computed with respect to the masks provided by $\mathcal{T}$, for a soccer match that is taken from the same competition as the training set. All curves reach approximately the same performances, with a slight advantage for adaptive networks produced by ARTHuS when initialized from $\mathcal{S}_{\text{pretrained}}$.

B.2.4 Tuning of the learning rate

As for any gradient-based learning algorithm, the learning rate may influence the results. Hereafter, we compare the results obtained with two learning rates when TinyNet is retrained online from a pre-trained network. As can be seen in Figure B.4, a higher learning rate rapidly leads to better performances but also involves much more drops of performances throughout the match. This is why we use the lower learning rate of 10^{-5} in the experiments of the main chapter.

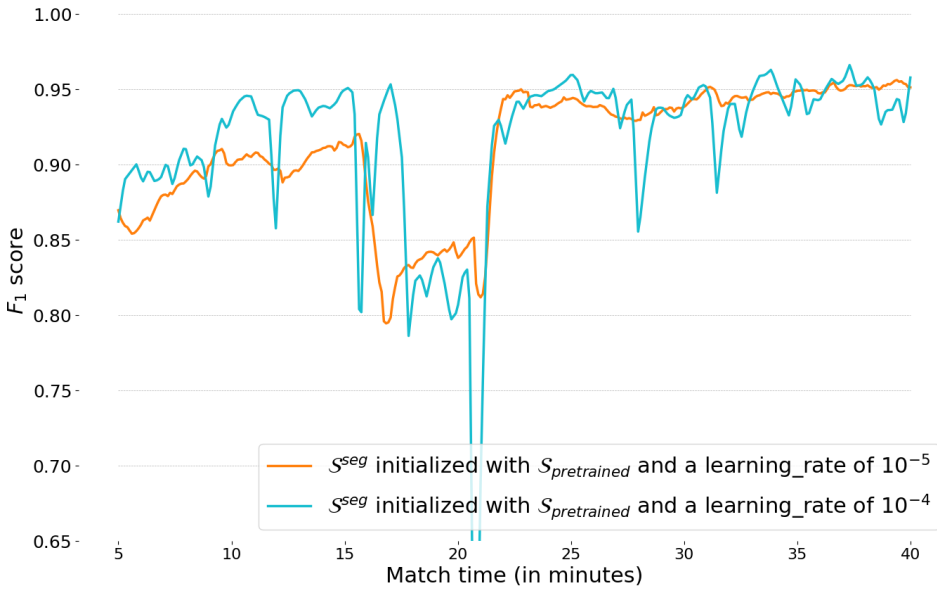


Figure B.4: Comparison of two different learning rates for ARTHuS when the network is trained online from a pre-trained network. A higher learning rate allows to reach better performances rapidly but is more prone to accidental drops of performances.

B.2.5 Other camera views

We also tested our method on another camera view. In this section, we show the result on one of the close-up cameras. This type of camera shows the players from a close-up point of view, which results in much bigger silhouettes compared to the main camera. Figure B.5 shows the result of our method when trained from scratch after 20 minutes of online training time. As can be seen, the players are still correctly segmented, which indicates that our method can be applied as is with different camera views. This is an encouraging result about the robustness of our method to various sports scenes. It also implies that there might be no need to manually annotate any frames, regardless of the camera, in order to have a working multi-camera system for human segmentation in sports.

Figure B.5: ARTHuS for close-up cameras. Results of our method on another soccer camera obtained with TinyNet trained from scratch.

APPENDIX C

Context-aware loss function: additional details and experiments

Contents

This appendix provides further details on the method presented in Chapter 6. In particular, Section C.1 describes the proposed iterative one-to-one matching. Section C.2 provides the details of the time-shift encoding between two consecutive event. Then, Section C.3 provides some extra analysis and results. Finally, Section C.4 shows additional qualitative results.

C.1 One-to-one matching

The iterative one-to-one matching between the predicted locations $\hat{\mathbf{Y}}_{\cdot,2}$ and the ground-truth locations $\mathbf{Y}_{\cdot,2}$ described in Chapter 6 is illustrated in Figure C.1. It is further detailed formally in Algorithm 2.

C.2 Additional details on the Time-Shift Encoding (TSE)

The time-shift encoding (TSE) described in Chapter 6 is further detailed below. We note $s^c(x)$ the TSE of frame x related to class c.

We denote s_p^c (resp. s_f^c) the difference between the frame index of x and the frame index of its closest past (resp. future) ground-truth action of class c. They constitute the time-shifts of x from its closest past and future ground-truth actions of class c, expressed

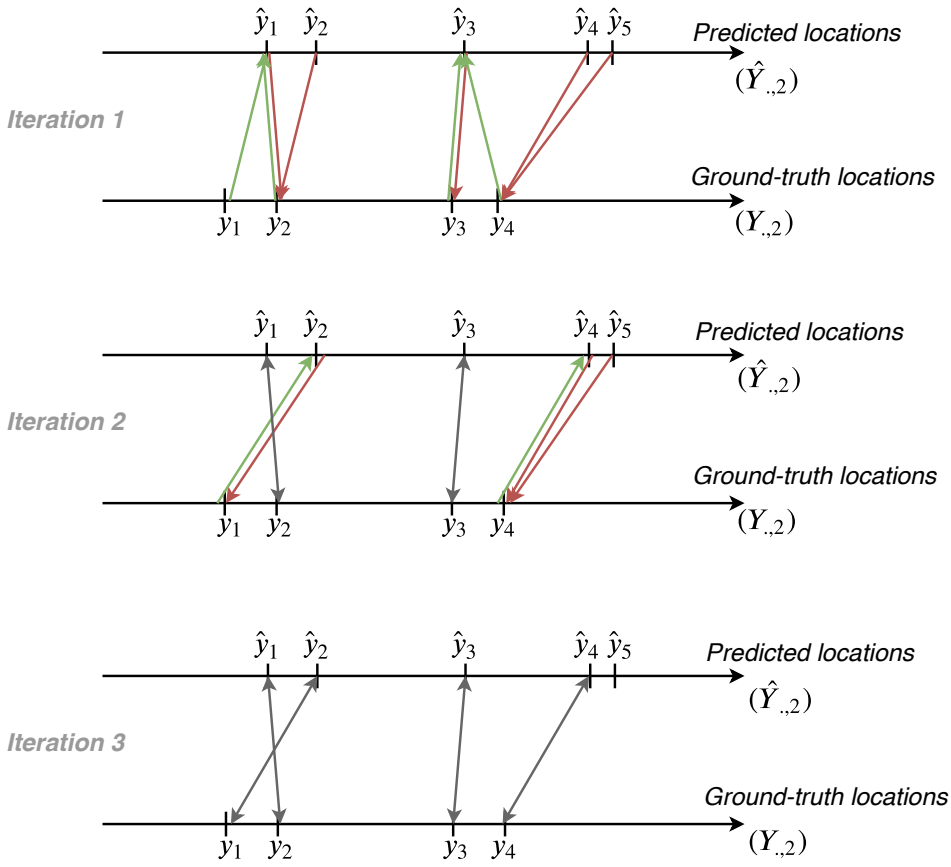


Figure C.1: Iterative one-to-one matching. Example of the iterative one-to-one matching. At iteration 1, each ground-truth location is matched with its closest predicted location (**green arrows**), and vice-versa (**brown arrows**). Locations that match each other are permanently matched (**gray arrows**), and the process is repeated with the remaining locations at iteration 2. In this case, two iterations suffice to match all the ground-truth locations with a predicted location, as evidenced by the absence of available ground-truth location for iteration 3.

Algorithm 2 : Iterative matching between ground-truth and predicted locations.

Data : $Y, \hat{Y}$ ground-truth and predicted locations
Result : Results: Matching couples $(y, \hat{y}) \in Y \times \hat{Y}$
Algorithm:
while $Y \neq \emptyset$ **do**
  $f : Y \to \hat{Y} : f(y) = \text{argmin}\{|y - \hat{y}| : \hat{y} \in \hat{Y}\}$;
  for $\hat{y} \in \hat{Y}$ **do**
    if $|f^{-1}(\{\hat{y}\})| \geq 1$ **then**
      $y_{\hat{y}} = \text{argmin}\{|y - \hat{y}| : y \in f^{-1}(\{\hat{y}\})\}$;
      Save matching couple $(y_{\hat{y}}, \hat{y})$;
      Remove $y_{\hat{y}}$ from Y and $\hat{y}$ from $\hat{Y}$;
    end if
  end for
end while

in number of frames (*i.e.* if frames 9 and 42 are actions of class c, then frame 29 has $s_p^c = 29 - 9 = 20$ and $s_f^c = 29 - 42 = -13$). We set $s_p^c = 0$ for a frame corresponding to a ground-truth action of class c, thus ensuring the relations $s_f^c < 0 \leq s_p^c$. The TSE $s^c(x)$ is defined as the time-shift among $\{s_p^c, s_f^c\}$ related to the action that has the dominant influence on x. The rules used to determine which time-shift is selected are the following:

- if $s_p^c < K_3^c$: keep s_p^c, because x is located *just after* the past action, which still strongly influences x.
- if $K_3^c \leq s_p^c < K_4^c$: x is in the *transition zone* after the past action, whose influence weakens, thus the decision depends on how far away is the future action:
  - if $s_f^c \leq K_1^c$: keep s_p^c, because x is located *far before* the future action, which does not yet influence x.
  - if $s_f^c > K_1^c$: The future action may be close enough to influence x:
    - if $\frac{s_p^c - K_3^c}{K_4^c - K_3^c} < \frac{K_2^c - s_f^c}{K_2^c - K_1^c}$: keep s_p^c, because x is closer to the *just after* region of the past action than it is to the *just before* region of the future action, with respect to the size of the transition zones.
    - else: keep s_f^c, because the future action influences x more than the past action.
- if $s_p^c \geq K_4^c$: keep s_f^c, because x is located *far after* the past action, which does not influence x anymore.

For completeness, let us recall the following details mentioned in Chapter 6. If x is both located *far after* the past action and *far before* the future action, selecting either of the two

time-shifts has the same effect in our loss. Furthermore, for the frames located either before the first or after the last annotated action of class c, only one time-shift can be computed and is thus set as $s^c(x)$. Finally, if no action of class c is present in the video, then we set $s^c(x) = K_1^c$ for all the frames. This induces the same behavior in our loss as if they were all located far before their closest future action.

The TSE is used to shape our novel context-aware loss function for the temporal segmentation module. The cases described above ensure the temporal continuity of the loss, regardless of the proximity between two actions of the same class, excepted at frames annotated as ground-truth actions. This temporal continuity can be visualized in Figure C.2, which shows a representation of $\tilde{L}(p,s)$ (analogous to Figure 6.1) when two actions are close to each other.

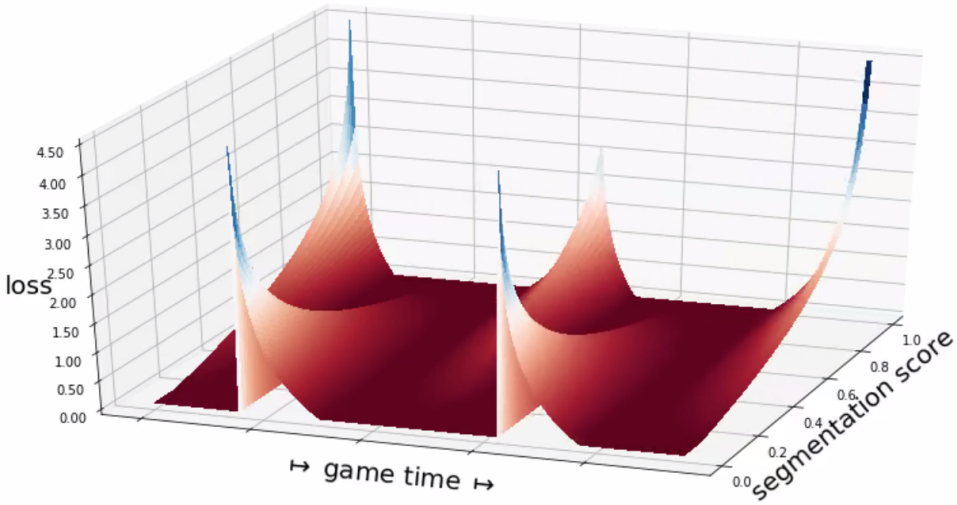


Figure C.2: Context-aware loss function (close actions). Representation of our segmentation loss when two actions of the same class are close to each other. The loss is parameterized by the time-shift encoding of the frames and is continuous through time, except at frames annotated as actions.

C.3 Extra analyses

C.3.1 Per-class results

As for the class *goal* in Figure 6.6, Figures C.3 and C.4 display the number of TP, FP, FN and the precision, recall and F_1 metrics for the classes *card* and *substitution* as a function

of the tolerance δ allowed for the localization of the spots.

Figure C.3 shows that most cards can be efficiently spotted by our model within 15 seconds around the ground truth (δ = 30 seconds). We achieve a precision of 66% for that tolerance. The previous baseline plateaus within 20 seconds (δ = 40 seconds) and still has a lower performance.

Figure C.4 shows that most substitutions can be efficiently spotted by our model within 15 seconds around the ground truth (δ = 30 seconds). We achieve a precision of 73% for that tolerance. The previous baseline reaches a similar performance for that tolerance, and reaches 82% within 60 seconds (δ = 120 seconds) around the ground truth.

Except for the precision metric for the substitutions with tolerances larger than 20 seconds, our model outperforms the previous baseline of SoccerNet. As mentioned in Chapter 6, for goals, many visual cues facilitate their spotting, *e.g.*, multiple replays, particular camera views, or celebrations from the players and from the public. Cards and substitutions are more difficult to spot since the moment the referee shows a player a card and the moment a new player enters the field to replace another are rarely replayed (*e.g.*, for cards, the foul is replayed, not the sanction). Also, the number of visual cues that allow their identification is reduced, as these actions generally do not lead to celebrations from the players or the public. Besides, cards and substitutions may not be broadcast in full screen, as they are sometimes merely shown from the main camera and are thus barely visible. Finally, substitutions occurring during the half-time are practically impossible to spot, as said in Chapter 6.

C.3.2 Segmentation loss analysis

We provide a supplementary analysis on the λ^{seg} parameter, which balances the segmentation loss and the action spotting loss in Equation 6.5 of Chapter 6. We fix different values of λ^{seg} and train a network for each value. We show the segmentation scores on one game for the *goal* class in Figure C.5. We also display the Average-mAP for the whole test set for the different values of λ^{seg}.

It appears that extreme values of λ^{seg} substantially influence both the action spotting performance and the segmentation curves, hence the automatic highlights generation. Small values (*i.e.* $\lambda^{\text{seg}} \leq 0.1$) produce a useless segmentation for spotting the interesting unannotated *goal opportunities*. This is because the loss does not provide a sufficiently strong feedback for the segmentation task as it does not penalize enough the segmentation scores. These values of λ^{seg} also lead to a decrease in the Average-mAP for the action spotting task, as already observed in the ablation study presented in Chapter 6. Moreover, very large values ($\lambda^{\text{seg}} \geq 100$) penalize too much the unannotated goal opportunities, for which the network is then forced to output very small segmentation scores. Such actions are thus more difficult to retrieve for the production of highlights. These values of λ^{seg} also lead to a large decrease in the Average-mAP for the action spotting

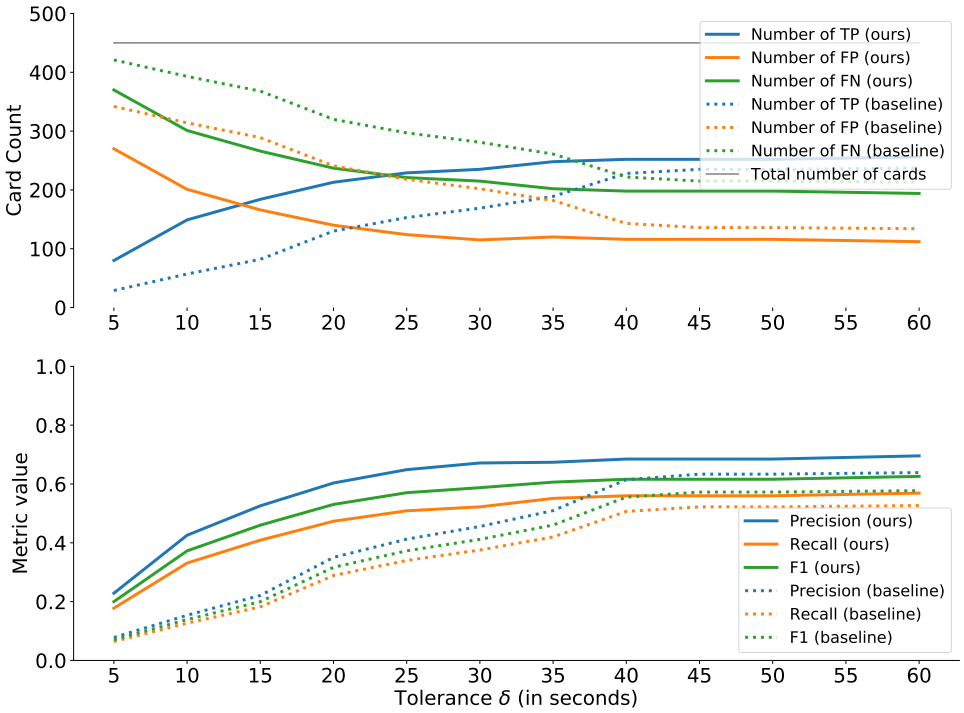


Figure C.3: Per-class results (cards). A prediction of class *card* is a **true positive (TP)** with tolerance δ when it is located at most $\delta/2$ seconds from a ground-truth card. The baseline results are obtained from the best model of [72]. Our model spots most cards within 15 seconds around the ground truth ($\delta = 30$ seconds).

task, as the feedback of the segmentation loss overshadows the feedback of the spotting loss. Finally, it seems that for $\lambda^{\text{seg}} \in [1, 10]$, the spotting performance is high while providing informative segmentation scores on *goal opportunities*. These values lead to the spotting of several *goal opportunities*, shown in Figure C.5, which might be included in the highlights automatically generated for this match by the method described in Chapter 6.

C.3.3 Comments on the improvements on ActivityNet

In Table 6.3, we report the averages over samples of 20 results for each metric, that we further analyze statistically below for the Av.-mAP. First, following D'Agostino's normality test, we can reasonably assume that the samples are normally distributed, since we obtain p-values > 0.1 (0.28 for BMN and 0.24 for ours respectively). The standard devi-

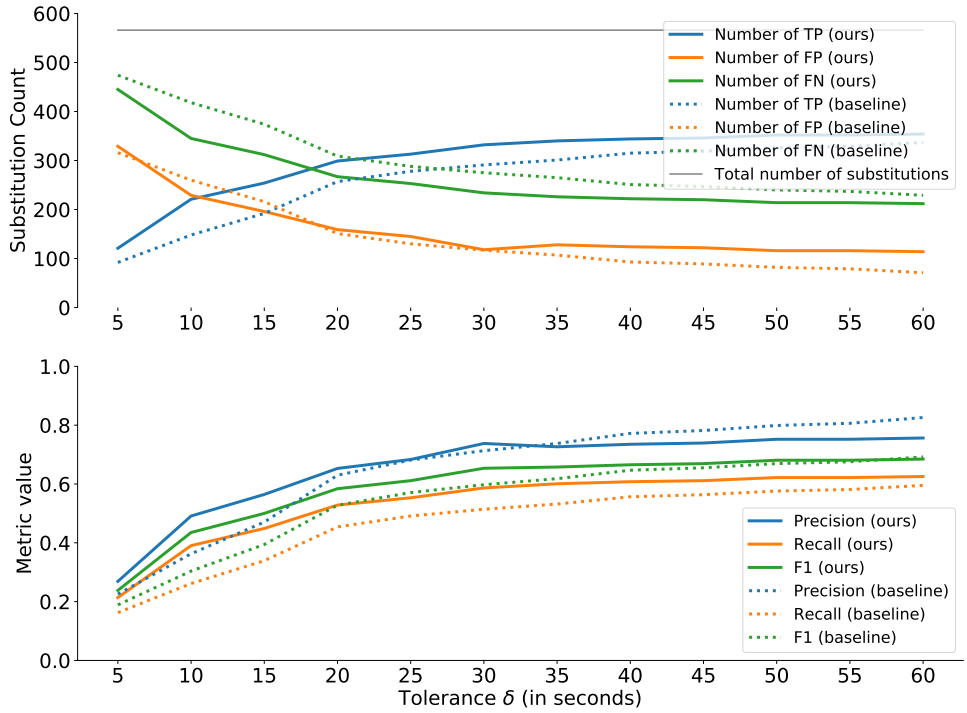


Figure C.4: Per-class results (substitutions). A prediction of class *substitution* is a **true positive (TP)** with tolerance δ when it is located at most $\delta/2$ seconds from a ground-truth substitution. The baseline results are obtained from the best model of [72]. Our model spots most substitutions within 15 seconds around the ground truth ($\delta = 30$ seconds).

ations of the samples are 0.08% and 0.07%. Since the difference between the averages is 0.38%, the normal distributions overlap beyond two standard deviations from their centers, which shows that our improvements are beyond noise domain. Furthermore, Bartlett's test for equal variances gives a p-value of 0.62 (> 0.1), which allows us to use Student's t-test to check whether the two samples can be assumed to have the same mean or not. We obtain a p-value of 2.3×10^{-18}, which strongly indicates that our results are significantly different from those of BMN and hence confirm the significant improvement. For the AR@100 and AUC, similar analyses give final p-values of 7.4×10^{-3} and 9.8×10^{-2}, which corroborates the statistical significance of our improvements.

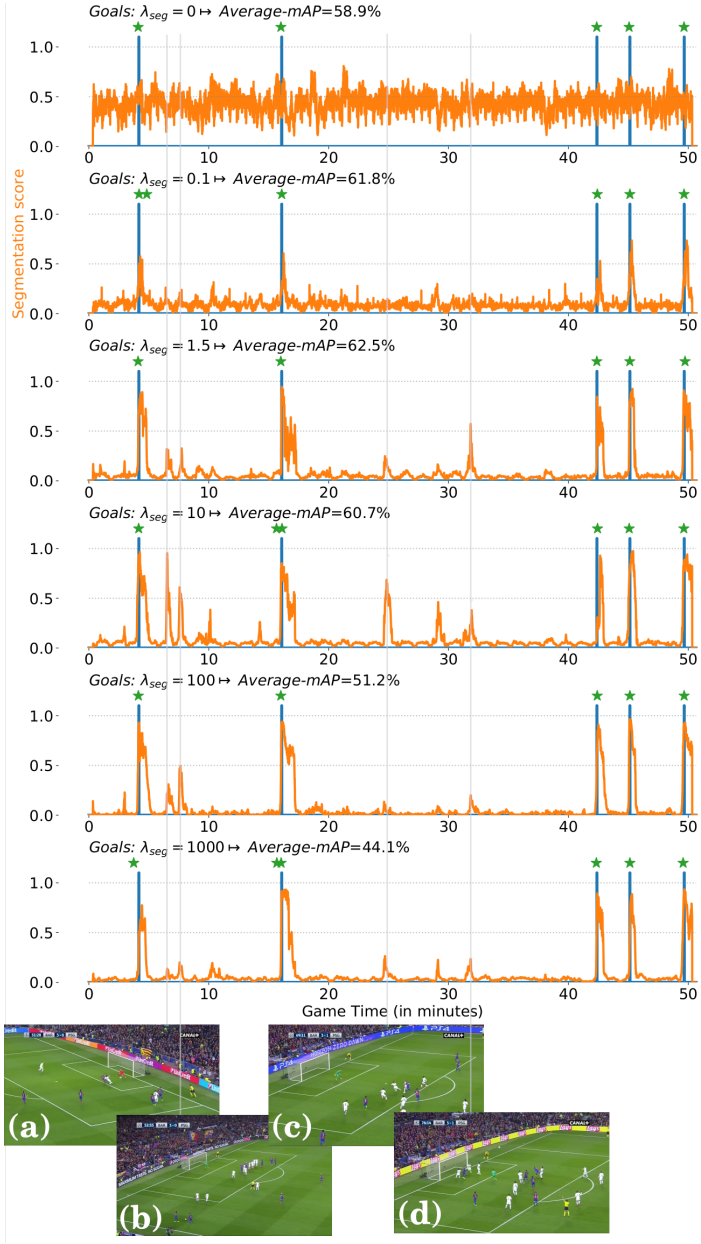


Figure C.5: Influence of λ^{seg} on the segmentation and spotting results of the second half of the famous *Remuntad* match, Barcelona - PSG, for the class *goal*, for different values of λ^{seg}. The best Average-mAP for the spotting task is located around $\lambda^{\text{seg}} = 1.5$, while the best value for spotting unannotated goal opportunities might be around $\lambda^{\text{seg}} = 10$. For this value, several meaningful *goal opportunities* have a high **segmentation score**: **(a)** a shot on a goal post, **(b)** a free kick, **(c)** lots of dribbles in the rectangle, and **(d)** a headshot right above the goal.

C.4 Extra actions and highlights generation

Figure C.6 shows additional action spotting and segmentation results. We can identify actions that are unannotated but display high segmentation scores such as goal opportunities and unsanctioned fouls. A goal opportunity around the 29^{th} minute can be identified through the segmentation results. Besides, a false positive spot (green star) for a card is predicted around the 9^{th} minute, further supported by a high segmentation score. A manual inspection reveals that a severe unsanctioned foul occurs at this moment. The automatic highlights generator presented in Chapter 6 would include it in the summary of the match. Even though this foul does not lead to a card for the offender, the content of this sequence corresponds to an interesting action that would be tolerable in a highlights video.

Figure C.7 shows a frame for which our network provides a high segmentation score and a false positive spot around the 26^{th} minute (*i.e.* 71^{st} minute of the match) for *substitutions* in Figure 6.7 of Chapter 6. We can see that the LED panel used by the referee to announce substitutions is visible on the frame. This may indicate that the network learns, quite rightly, to associate this panel with substitutions. As a matter of fact, at this moment, even the commentator announces that a substitution is probably imminent.

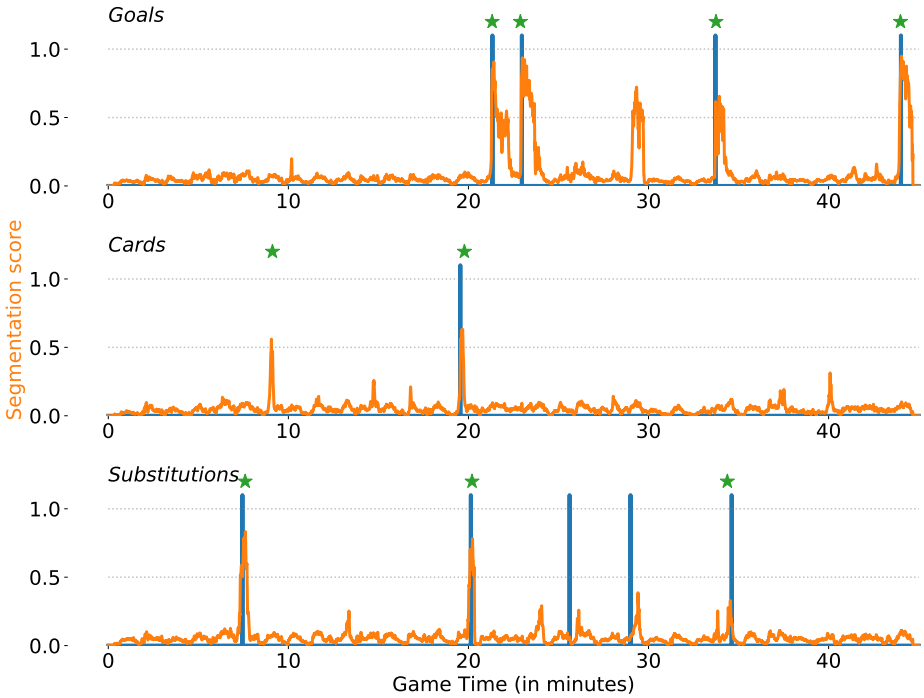


Figure C.6: Extra action spotting and segmentation results. These results are obtained on the second half of the match Barcelona - Espanyol in December 2016. **Ground truth actions**, **temporal segmentation curves**, and **spotting results (green stars)** are illustrated. Unannotated actions can be identified and included in the highlights using our segmentation. For example, a goal opportunity occurs around the 29^{th} minute. A false positive spot for a card is predicted by our network around the 9^{th} minute. As it corresponds to a severe unsanctioned foul, it is fine for our automatic highlights generator to include it in the summary of the match.

Figure C.7: False positive spot of a substitution for the second half of the famous *Remuntada* match, Barcelona - PSG, in March 2017. The LED panel used to announce substitutions is visible on the left, which presumably explains why the network predicted the sequence around this frame as a substitution.

Bibliography

[1] Bayesian optimization. https://github.com/fmfn/BayesianOptimization.

[2] Bayesian Optimization. https://github.com/fmfn/BayesianOptimization.

[3] Code for BMN. https://github.com/JJBOY/BMN-Boundary-Matching-Network.

[4] Google Scholar, top publication conferences and journals. https://scholar.google.com/citations?view_op=top_venues&hl=en.

[5] H. Alwassel, F. Caba Heilbron, V. Escorcia, and B. Ghanem. Diagnosing error in temporal action detectors. In *European Conference on Computer Vision (ECCV)*, pages 264–280, Munich, Germany, September 2018.

[6] H. Alwassel, F. Caba Heilbron, and B. Ghanem. Action search: Spotting targets in videos and its application to temporal action localization. In *European Conference on Computer Vision (ECCV)*, pages 253–269, Munich, Germany, September 2018.

[7] M. Archana and M. Kalaiselvi Geetha. An efficient ball and player detection in broadcast tennis video. *International Journal of Intelligent Systems Technologies and Applications*, 384:427–436, August 2016.

[8] M. Babaee, D. Dinh, and G. Rigoll. A deep convolutional neural network for background subtraction. *Pattern Recognition*, 76:635–649, April 2018.

[9] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Action classification in soccer videos with long short-term memory recurrent neural networks. In *International Conference on Artificial Neural Networks (ICANN)*, pages 154–159, Thessaloniki, Greece, September 2010.

[10] L. Baraldi, M. Douze, R. Cucchiara, and H. Jégou. LAMV: Learning to align and match videos with kernelized temporal layers. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7804–7813, Salt Lake City, Utah, USA, June 2018.

[11] O. Barnich and M. Van Droogenbroeck. Code for ViBe. https://orbi.uliege.be/handle/2268/145853.

[12] O. Barnich and M. Van Droogenbroeck. ViBe: a powerful random technique to estimate the background in video sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 945–948, Taipei, Taiwan, April 2009.

[13] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, June 2011.

[14] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari. 360° Detection and tracking algorithm of both pedestrian and vehicle using fisheye images. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 132–137, Las Vegas, Nevada, USA, June 2015.

[15] V. Bettadapura, C. Pantofaru, and I. Essa. Leveraging contextual cues for generating basketball highlights. In *ACM international conference on Multimedia (ACM-MM)*, pages 908–917, Amsterdam, The Netherlands, October 2016.

[16] S. Bianco, G. Ciocca, and R. Schettini. Combination of video change detection algorithms by genetic programming. *IEEE Transactions on Evolutionary Computation*, 21(6):914–928, December 2017.

[17] J. Bouguet. Camera calibration toolbox for Matlab, 2014.

[18] T. Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer Science Review*, 11-12:31–66, May 2014.

[19] T. Bouwmans and B. Garcia-Garcia. Background subtraction in real applications: Challenges, current models and future directions. *CoRR*, abs/1901.03577, January 2019.

[20] K. Boyd, V. Costa, J. Davis, and C. Page. Unachievable region in precision-recall space and its effect on empirical evaluation. In *International Conference on Machine Learning (ICML)*, pages 639–646, Edinburgh, UK, June-July 2012.

[21] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[22] M. Braham, S. Piérard, and M. Van Droogenbroeck. Semantic background subtraction. In *IEEE International Conference on Image Processing (ICIP)*, pages 4552–4556, Beijing, China, September 2017.

[23] M. Braham and M. Van Droogenbroeck. Deep background subtraction with scene-specific convolutional neural networks. In *IEEE International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–4, Bratislava, Slovakia, May 2016.

[24] L. Bridgeman, M. Volino, J. Guillemaut, and A. Hilton. Multi-person 3D pose estimation and tracking in sports. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2487–2496, Long Beach, California, USA, June 2019.

[25] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *British Machine Vision Conference (BMVC)*, pages 93.1–93.12, London, United Kingdom, September 2017.

[26] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. Niebles. SST: Single-Stream Temporal Action Proposals. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6373–6382, Honolulu, Hawaii, USA, July 2017.

[27] C. Bucila, R. Caruana, and A. Niculescu-Mizil. Model compression. In *ACM SIGKDD International conference on Knowledge discovery and data mining (KDD)*, pages 535–541, Philadelphia, Pennsylvania, USA, August 2006.

[28] M. Buric, M. Ivasic-Kos, and M. Pobar. Player tracking in sports videos. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 334–340, Sydney, Australia, December 2019.

[29] Z. Cai, H. Neher, K. Vats, D. Clausi, and J. Zelek. Temporal hockey action recognition via pose and optical flows. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2543–2552, Long Beach, California, USA, June 2019.

[30] J. Carreira, E. Noland, C. Hillier, and A. Zisserman. A short note on the kinetics-700 human action dataset. *CoRR*, abs/1907.06987, July 2019.

[31] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, Honolulu, Hawaii, USA, July 2017.

[32] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, volume 11211 of *Lecture Notes in Computer Science*, pages 801–818, Munich, Germany, September 2018. Springer.

[33] A. Cioppa, M. Braham, and M. Van Droogenbroeck. Asynchronous semantic background subtraction. *Journal of Imaging*, 6(6):1–20, June 2020.

[34] A. Cioppa, A. Deliège, S. Giancola, B. Ghanem, M. Van Droogenbroeck, R. Gade, and T. Moeslund. A context-aware loss function for action spotting in soccer videos. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13123–13133, Seattle, Washington, USA, June 2020.

[35] A. Cioppa, A. Deliège, M. Istasse, C. De Vlesschouwer, and M. Van Droogenbroeck. ARTHuS: Adaptive real-time human segmentation in sports through online distillation. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2505–2514, Long Beach, California, USA, June 2019.

[36] A. Cioppa, A. Deliège, N. Ul Huda, R. Gade, M. Van Droogenbroeck, and T. Moeslund. Multimodal and multiview distillation for real-time player detection on a football field. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 3846–3855, Seattle, Washington, USA, June 2020.

[37] A. Cioppa, A. Deliège, and M. Van Droogenbroeck. A bottom-up approach based on semantics for the interpretation of the main camera stream in soccer games. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 1846–1855, Salt Lake City, Utah, USA, June 2018.

[38] A. Cioppa, M. Van Droogenbroeck, and M. Braham. Real-time semantic background subtraction. In *IEEE International Conference on Image Processing (ICIP)*, pages 3214–3218, Abu Dhabi, United Arab Emirates, October 2020.

[39] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, Las Vegas, Nevada, USA, June 2016.

[40] C. Cuevas, E. Yanez, and N. Garcia. Labeled dataset for integral evaluation of moving object detection algorithms: LASIESTA. *Computer Vision and Image Understanding*, 152:103–117, November 2016.

[41] C. Dai, Y. Zheng, and X. Li. Layered representation for pedestrian detection and tracking in infrared imagery. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 13–13, San Diego, California, USA, June 2005.

[42] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, San Diego, California, USA, June 2005.

[43] D. Damen, H. Doughty, G. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The EPIC-KITCHENS dataset. *CoRR*, abs/1804.02748, April 2018.

[44] D. Damen, H. Doughty, G. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. The EPIC-KITCHENS dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 2020.

[45] A. Deliège, A. Cioppa, and M. Van Droogenbroeck. HitNet: a neural network with capsules embedded in a Hit-or-Miss layer, extended with hybrid data augmentation and ghost capsules. *CoRR*, abs/1806.06519, June 2018.

[46] A. Deliège, A. Cioppa, and M. Van Droogenbroeck. An effective hit-or-miss layer favoring feature interpretation as learned prototypes deformations. In *AAAI Conference on Artificial Intelligence, Workshop on Network Interpretability for Deep Learning*, pages 1–8, Honolulu, Hawaii, USA, January-February 2019.

[47] A. Deliège, A. Cioppa, and M. Van Droogenbroeck. Ghost loss to question the reliability of training data. *IEEE Access*, 8:44774–44782, March 2020.

[48] Deloitte. Market size of the European football market from 2006/07 to 2015/16 (in billion euros), 2017.

[49] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, Florida, USA, June 2009.

[50] T. D'Orazio, M. Leo, P. Spagnolo, P. Mazzeo, N. Mosca, M. Nitti, and A. Distante. An investigation into the feasibility of real-time soccer offside detection from a multiple camera system. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(12):1804–1818, December 2009.

[51] T. D'Orazio, M. Leo, P. Spagnolo, M. Nitti, N. Mosca, and A. Distante. A visual system for real time detection of goal events during soccer matches. *Computer Vision and Image Understanding*, 113(5):622–632, May 2009.

[52] S. Ebadi and E. Izquierdo. Foreground segmentation with tree-structured sparse RPCA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(9):2273–2280, September 2018.

[53] A. Ekin, A. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, July 2003.

[54] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision (ECCV)*, volume 1843 of *Lecture Notes in Computer Science*, pages 751–767. Springer, June 2000.

[55] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes challenge 2012 (VOC2012) results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[56] D. Farin, S. Krabbe, and W. Effelsberg et.al. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia*, volume 5307 of *Proceedings of SPIE*, pages 80–92, San Jose, California, USA, December 2003.

[57] H. Faulkner and A. Dick. AFL player detection and tracking. In *Digital Image Computing: Techniques and Applications*, pages 1–8, Adelaide, Australia, November 2015.

[58] P. Felsen, P. Agrawal, and J. Malik. What will happen next? forecasting player moves in sports videos. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3362–3371, Venice, Italy, October 2017.

[59] P. Figueroa, N. Leite, and R. Barros. Tracking soccer players aiming their kinematical motion analysis. *Computer Vision and Image Understanding*, 101(2):122–135, February 2006.

[60] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM: Graphics and Image Processing*, 24(6):381–395, June 1981.

[61] A. Fitzgibbon and R. Fisher. A buyer's guide to conic fitting. In *British Machine Vision Conference (BMVC)*, pages 513–522, Birmingham, United Kingdom, September 1995.

[62] M. Fonder and M. Van Droogenbroeck. Mid-Air: A multi-modal dataset for extremely low altitude drone flights. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), UAVision*, pages 553–562, Long Beach, California, USA, June 2019.

[63] T. Furlanello, Z.C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. In *International Conference on Machine Learning (ICML)*, volume 80, pages 1607–1616, Stockholm, Sweden, July 2018.

[64] R. Gade, A. Jørgensen, and T. Moeslund. Occupancy analysis of sports arenas using thermal imaging. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 277–283, Rome, Italy, February 2012. SCITEPRESS Digital Library.

[65] R. Gade, A. Jørgensen, and T. Moeslund. Long-term occupancy analysis using graph-based optimisation in thermal imagery. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3698–3705, Portland, Oregon, USA, June 2013. IEEE Computer Society Press.

[66] R. Gade and T. Moeslund. Thermal cameras and applications: A survey. *IAPR Workshop on Machine Vision Applications (MVA)*, 25(1):245–262, October 2014.

[67] R. Gade and T. Moeslund. Constrained multi-target tracking for team sports activities. *IPSJ Transactions on Computer Vision and Applications*, 10(1):1–11, January 2018.

[68] J. Gao, K. Chen, and R. Nevatia. CTAP: Complementary Temporal Action Proposal Generation. In *European Conference on Computer Vision (ECCV)*, pages 70–85, Munich, Germany, September 2018.

[69] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia. TURN TAP: Temporal unit regression network for temporal action proposals. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3648–3656, Venice, Italy, October 2017.

[70] X. Gao, Z. Niu, D. Tao, and X. Li. Non-goal scene analysis for soccer video. *Neurocomputing*, 74(4):540–548, January 2011.

[71] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, April 2017.

[72] S. Giancola, M. Amine, T. Dghaily, and B. Ghanem. SoccerNet: A scalable dataset for action spotting in soccer videos. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1711–1721, Salt Lake City, Utah, USA, June 2018.

[73] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, Columbus, Ohio, USA, June 2014.

[74] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6047–6056, Salt Lake City, Utah, USA, June 2018.

[75] A. Gupta, J. Little, and R. Woodham. Using line and ellipse features for rectification of broadcast hockey video. In *Canadian Conference on Computer and Robot Vision (CRV)*, pages 32–39, St. Johns, Canada, May 2011.

[76] S. h. Lee, G. c. Lee, J. Yoo, and S. Kwon. WisenetMD: Motion detection using dynamic background region analysis. *Symmetry*, 11(5):1–15, May 2019.

[77] T. Halperin, A. Ephrat, and S. Peleg. Dynamic temporal alignment of speech to lips. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3980–3984, Brighton, United Kingdom, May 2019.

[78] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, March 2018.

[79] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, December 2015.

[80] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, Nevada, USA, June 2016.

[81] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Niebles. ActivityNet: A large-scale video benchmark for human activity understanding. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970, Boston, Massachusetts, USA, June 2015.

[82] D. Heo, E. Lee, and B. Chul Ko. Pedestrian detection at night using deep neural networks and saliency maps. *Journal of Imaging Science and Technology*, 61(6):60403–1–60403–9, January 2017.

[83] C. Herrmann, T. Müller, D. Willersinn, and J. Beyerer. Real-time person detection in low-resolution thermal infrared imagery with MSER and CNNs. In *Security + Defence (SPIE)*, page 99870I, Edinburgh, United Kingdom, September 2016.

[84] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, March 2015.

[85] N. Homayounfar, S. Fidler, and R. Urtasun. Sports field localization via deep structured models. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4012–4020, Honolulu, Hawaii, USA, July 2017.

[86] M. Beetzand N. Hoyningen-Huene, B. Kirchlechner, S. Gedikli, F. Silesand, M. Durus, and M. Lames. Aspogamo: Automated sports game analysis models. *International Journal of Computer Science in Sport*, 8(1):1–21, 2009.

[87] C. Huang, H. Shih, and C. Chao. Semantic analysis of soccer video using dynamic bayesian network. *IEEE Transactions on Multimedia*, 8(4):749–760, August 2006.

[88] Y. Huang, J. Llach, and S. Bhagavathy. Players and ball detection in soccer videos based on color segmentation and shape analysis. In *Multimedia Content Analysis and Mining*, volume 4577 of *Lecture Notes in Computer Science*, pages 416–425. Springer, 2007.

[89] N. Huda, B. Hansen, R. Gade, and T. Moeslund. Occupancy analysis of soccer fields using wide-angle lens. In *International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 354–359, Jaipur, India, December 2017.

[90] N. Huda, K. Jensen, R. Gade, and T. Moeslund. Estimating the number of soccer players using simulation-based occlusion handling. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1937–1946, Salt Lake City, Utah, USA, June 2018.

[91] V. Iashin and E. Rahtu. A better use of audio-visual cues: Dense video captioning with bi-modal transformer. In *British Machine Vision Conference (BMVC)*, Remotely-held conference, September 2020.

[92] V. Iashin and E. Rahtu. Multi-modal dense video captioning. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4117–4126, Seattle, Washington, USA, June 2020.

[93] International Football Association Board (IFAB). Laws of the game 20/21. https://static-3eb8.kxcdn.com/files/document-category/062020/fXHLhQuMmtekmfe.pdf.

[94] K. Ingersoll. Vision based multiple target tracking using recursive RANSAC. Master's thesis, Brigham Young University, March 2015.

[95] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, Lille, France, July 2015.

[96] M. Isogawa, D. Mikami, K. Takahashi, D. Iwai, K. Sato, and H. Kimata. Which is the better inpainted image? training data generation without any manual operations. *International Journal of Computer Vision*, 127:1751–1766, November 2019.

[97] M. Istasse, J. Moureau, and C. De Vleesschouwer. Associative embedding for team discrimination. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 2477–2486, Long Beach, California, USA, June 2019.

[98] Z. Ivankovic, B. Markoski, M. Ivkovic, D. Radosav, and P. Pecev. AdaBoost in basketball player identification. In *IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 151–156, Budapest, Hungary, November 2012.

[99] S. Javed, A. Mahmood, T. Bouwmans, and S. K. Jung. Background-foreground modeling based on spatiotemporal sparse subspace clustering. *IEEE Transactions on Image Processing*, 26(12):5840–5854, December 2017.

[100] H. Jiang, Y. Lu, and J. Xue. Automatic soccer video event detection based on a deep neural network combined CNN and RNN. In *International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 490–494, San Jose, California, USA, November 2016.

[101] S. Jiang and X. Lu. WeSamBE: A weight-sample-based method for background subtraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2105–2115, September 2018.

[102] Y. Jiang, K. Cui, L. Chen, C. Wang, and C. Xu. SoccerDB: A large-scale database for comprehensive video understanding. *CoRR*, abs/1912.04465, December 2019.

[103] Y. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS Challenge: Action Recognition with a Large Number of Classes. http://crcv.ucf.edu/THUMOS14/, 2014.

[104] P-M. Jodoin and J. Konrad. Change detection website. http://changedetection.net/.

[105] Y. Kang, J. Lim, Q. Tian, and M. Kankanhalli. Soccer video event detection with visual keywords. In *Joint Conference of the International Conference on Information, Communications and Signal Processing, and Pacific Rim Conference on Multimedia*, volume 3, pages 1796–1800, Singapore, December 2003.

[106] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, Colombus, Ohio, USA, June 2014.

[107] A.T. Kearney. Global sports market - total revenue from 2005 to 2017 (in billion U.S. dollars), 2014.

[108] A. Khan, B. Lazzerini, G. Calabrese, and L. Serafini. Soccer event detection. In *International Conference on Image Processing and Pattern Recognition (IPPR)*, pages 119–129, Copenhagen, Denmark, April 2018.

[109] H. Kim, J. Jung, and J. Paik. Fisheye lens camera based surveillance system for wide field of view monitoring. *Optik – International Journal for Light and Electron Optics*, 127(14):5636–5646, July 2016.

[110] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, December 2014.

[111] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, California, USA, May 2015.

[112] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9404–9413, Long Beach, CA, USA, June 2019.

[113] A. Kirillov, Y. Wu, K. He, and R. Girshick. PointRend: Image segmentation as rendering. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9796–9805, Seattle, Washington, USA, June 2020.

[114] A. Kolchinsky and DH. Wolpert. Semantic information, autonomous agency and non-equilibrium statistical physics. *Interface Focus*, 8(6), October 2018.

[115] D. Levi and S. Silberstein. Tracking and motion cues for rear-view pedestrian detection. In *International Conference on Intelligent Transportation Systems (ITSC)*, pages 664–671, Las Palmas de Gran Canaria, Spain, September 2015.

[116] A. Li, M. Thotakuri, D. Ross, J. Carreira, A. Vostrikov, and A. Zisserman. The AVA-kinetics localized human actions video dataset. *CoRR*, abs/2005.00214, May 2020.

[117] W. Li, D. Zheng, T. Zhao, and M. Yang. An effective approach to pedestrian detection in thermal imagery. In *International Conference on Natural Computation*, pages 325–329, Orléans, France, May 2012.

[118] D. Liang, Y. Liu, Q. Huang, and W. Gao. A scheme for ball detection and tracking in broadcast soccer video. In *Pacific Rim Conference on Multimedia (PCM)*, pages 864–875, Jeju Island, Korea, November 2005.

[119] L. Lim and H. Keles. Foreground segmentation using convolutional neural networks for multiscale feature encoding. *Pattern Recognition Letters*, 112:256–262, September 2018.

[120] T. Lin, X. Liu, X. Li, E. Ding, and S. Wen. BMN: Boundary-Matching Network for temporal action proposal generation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3888–3897, Seoul, Korea, October 2019.

[121] T. Lin, X. Zhao, and Z. Shou. Temporal Convolution Based Action Proposal: Submission to ActivityNet 2017. *CoRR*, abs/1707.06750, July 2017.

[122] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang. BSN: Boundary Sensitive Network for temporal action proposal generation. In *European Conference on Computer Vision (ECCV)*, pages 3–21, Munich, Germany, September 2018.

[123] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision (ECCV)*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, September 2014.

[124] X. Liu, Z. Deng, and Y. Yang. Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52:1089–1106, June 2018.

[125] Y. Liu, L. Ma, Y. Zhang, W. Liu, and S. Chang. Multi-granularity generator for temporal action proposal. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3599–3608, Long Beach, California, USA, June 2019.

[126] F. Long, T. Yao, Z. Qiu, X. Tian, J. Luo, and T. Mei. Gaussian temporal awareness networks for action localization. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 344–353, Long Beach, California, USA, June 2019.

[127] J. Lu, J. Chen, and J. Little. Pan-tilt-zoom SLAM for sports videos. In *British Machine Vision Conference (BMVC)*, pages 1–14, Cardiff, Wales, September 2019.

[128] M. Sah and C. Direkoglu. Evaluation of image representations for player detection in field sports using convolutional neural networks. In *International Conference on Theory and Application of Fuzzy Systems and Soft Computing (ICAFS)*, pages 107–115, Warsaw, Poland, August 2018. Springer International Publishing.

[129] L. Maddalena and A. Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168–1177, July 2008.

[130] Z. Mahmood, T. Ali, and S. Khattak. Automatic player detection and recognition in images using AdaBoost. In *International Bhurban Conference on Applied Sciences Technology (IBCAST)*, pages 64–69, Islamabad, Pakistan, January 2012.

[131] E. Malis and M. Vargas. Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA, 2007.

[132] M. Manafifard, H. Ebadi, and H. Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding*, 159:19–46, June 2017.

[133] W. McNally, K. Vats, T. Pinto, C. Dulhanty, J. McPhee, and A. Wong. GolfDB: A video database for golf swing sequencing. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2553–2562, Long Beach, California, USA, June 2019.

[134] T. Moeslund, G. Thomas, and A. Hilton. *Computer vision in sports*. Springer, 2014.

[135] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–59, Salt Lake City, Utah, USA, June 2018.

[136] F. Mufti, R. Mahony, and J. Heinzmann. Robust estimation of planar surfaces using spatio-temporal RANSAC for applications in autonomous vehicle navigation. 60(1):16–28, January 2012.

[137] S. Nam, H. Kim, and J. Kim. Trajectory estimation based on globally consistent homography. In *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 214–221. Springer, August 2003.

[138] V. Nguyen, T. Nguyen, and S. Chung. ConvNets and AGMM based real-time human detection under fisheye camera for embedded surveillance. In *International Conference on Information and Communication Technology Convergence (ICTC)*, pages 840–845, Jeju Island, Korea, October 2016. IEEE.

[139] C. Palmero, A. Clapés, C. Bahnsen, A. Møgelmose, T. Moeslund, and S. Escalera. Multi-modal RGB-depth-thermal human body segmentation. *International Journal of Computer Vision*, 118(2):217–239, 2016.

[140] S. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[141] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, June 2000.

[142] P. Parisot and C. De Vleeschouwer. Scene-specific classifier for effective and efficient team sport players detection from a single calibrated camera. *Computer Vision and Image Understanding*, 159:74–88, June 2017.

[143] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, June 2017.

[144] S. Piérard and M. Van Droogenbroeck. Summarizing the performances of a background subtraction algorithm measured on several videos. In *IEEE International Conference on Image Processing (ICIP)*, pages 3234–3238, Abu Dhabi, United Arab Emirates, October 2020.

[145] AJ. Piergiovanni and M. Ryoo. Fine-grained activity recognition in baseball videos. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1821–1828, Salt Lake City, Utah, USA, June 2018.

[146] G. Pingali, A. Opalach, and Y. Jean. Ball tracking and virtual replays for innovative tennis broadcasts. In *IEEE International Conference on Pattern Recognition (ICPR)*, volume 4, pages 152–156, Barcelona, Spain, September 2000.

[147] P. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, pages 1990–1998, Montreal, Canada, December 2015.

[148] L. Pishchulin, A. Jain, M. Andriluka, T. Thormahlen, and B Schiele. Articulated people detection and pose estimation: Reshaping the future. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3178–3185, Providence, Rhode Island, USA, June 2012.

[149] Miran Pobar and Marina Ivasic-Kos. Mask R-CNN and optical flow based method for detection and marking of handball actions. In *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, Beijing, China, October 2018.

[150] Y. Qian and D. Lee. Adaptive field detection and localization in robot soccer. In *RoboCup 2016*, volume 9776 of *Lecture Notes in Computer Science*, pages 218–229. Springer, June-July 2016.

[151] T. Rahman, B. Xu, and L. Sigal. Watch, listen and tell: Multi-modal weakly supervised dense event captioning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 8907–8916, Seoul, Korea, October 2019.

[152] V. Ramanathan, J. Huang, S. Abu-El-Haija, A. Gorban, K. Murphy, and L. Fei-Fei. Detecting events and key actors in multi-person videos. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3043–3053, Las Vegas, Nevada, USA, June 2016.

[153] U. Rao and U. Pati. A novel algorithm for detection of soccer ball and player. In *International Conference on Communications and Signal Processing (ICCSP)*, pages 344–348, Chengdu, China, April 2015.

[154] N. Reddy, P. Singhal, and K. Krishna. Semantic motion segmentation using dense CRF formulation. In *Indian Conference on Computer Vision Graphics and Image Processing*, pages 1–8, Bangalore, India, December 2014.

[155] J. Redmon, S. Divvala, R. Gorshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, June 2016.

[156] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *CoRR*, abs/1804.02767, April 2018.

[157] R. Reede and J. Joemon. Football video segmentation based on video production strategy. In *European Conference on Information Retrieval (ECIR)*, pages 433–446, Santiago de Compostela, Spain, June 2005.

[158] K. Rematas, I. Kemelmacher-Shlizerman, B. Curless, and S. Seitz. Soccer on your tabletop. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4738–4747, Salt Lake City, Utah, USA, June 2018.

[159] V. Reno, N. Mosca, M. Nitti, T. Dorazio, D. Campagnoli, A. Prati, and E. Stella. Tennis player segmentation for semantic behavior analysis. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 718–725, Santiago, Chile, December 2015.

[160] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. FitNets: Hints for thin deep nets. *CoRR*, abs/1412.6550, December 2014.

[161] G. Ros, L. Sellart, J. Materzynska, D. Vázquez, and A. López. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243, Las Vegas, Nevada, USA, June 2016.

[162] S. Roy and A. Ghosh. Real-time adaptive Histogram Min-Max Bucket (HMMB) model for background subtraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(7):1513–1525, July 2018.

[163] S. Sabour, N. Frosst, and G. Hinton. Dynamic routing between capsules. *CoRR*, abs/1710.09829, October 2017.

[164] S. Safdarnejad, X. Liu, L. Udpa, B. Andrus, J. Wood, and D. Craven. Sports videos in the wild (SVW): A video dataset for sports analysis. In *IEEE International Conference on Automatic Face and Gesture Recognition (FGR) Workshops*, volume 1, pages 1–7, Ljubljana, Slovenia, May 2015.

[165] D. Sahoo, Q. Pham, J. Lu, and S.C.H. Hoi. Online deep learning: Learning deep neural networks on the fly. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2660–2666, Stockholm, Sweden, July 2018.

[166] M. Saito, K. Kitaguchi, G. Kimura, and M. Hashimoto. People detection and tracking from fish-eye image based on probabilistic appearance model. In *SICE Annual Conference*, pages 435–440, Tokyo, Japan, September 2011.

[167] M. Sanabria, Sherly, F. Precioso, and T. Menguy. A deep architecture for multimodal summarization of soccer games. In *ACM International Conference on Multimedia (ACM-MM) Workshops*, pages 16–24, Nice, France, October 2019.

[168] S. Sarkar, A. Chakrabarti, and D. Prasad Mukherjee. Generation of ball possession statistics in soccer using minimum-cost flow network. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2515–2523, Long Beach, California, USA, June 2019.

[169] 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 *International Conference on Image Analysis and Processing (ICIAP)*, pages 196–203, Florence, Italy, September 1997. Springer.

[170] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3889–3898, Las Vegas, Nevada, USA, June 2016.

[171] P. Shukla, H. Sadana, A. Bansal, D. Verma, C. Elmadjian, B. Raman, and M. Turk. Automatic cricket highlight generation using event-driven and excitement-based features. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1881–1888, Salt Lake City, Utah, USA, June 2018.

[172] G. Sigurdsson, O. Russakovsky, and A. Gupta. What actions are needed for understanding human actions in videos? In *IEEE International Conference on Computer Vision (ICCV)*, pages 2156–2165, Venice, Italy, October 2017.

[173] G. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision (ECCV)*, pages 510–526, Amsterdam, The Netherlands, October 2016.

[174] K. Soomro and A. Zamir. Action recognition in realistic sports videos. In *Computer Vision in Sports*. Springer, January 2015.

[175] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. A self-adjusting approach to change detection based on background word consensus. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 990–997, Waikoloa Beach, Hawaii, USA, January 2015.

[176] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. SuBSENSE: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, January 2015.

[177] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin. Universal background subtraction using word consensus models. *IEEE Transactions on Image Processing*, 25(10):4768–4781, October 2016.

[178] Statista. Computer vision artificial intelligence (AI) market revenues worldwide, from 2015 to 2019, by application (in million U.S. dollars), 2016.

[179] C. Stauffer and E. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 246–252, Fort Collins, Colorado, USA, June 1999.

[180] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, April 1985.

[181] M. Tavassolipour, M. Karimian, and S. Kasaei. Event detection and summarization in soccer videos using bayesian network and copula. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(2):291–304, February 2014.

[182] R. Theagarajan, F. Pala, X. Zhang, and B. Bhanu. Soccer: Who has the ball? generating visual analytics and player statistics. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1830–1838, Salt Lake City, Utah, USA, June 2018.

[183] G. Thomas, R. Gade, T. Moeslund, P. Carr, and A. Hilton. Computer vision for sports: current applications and research topics. *Computer Vision and Image Understanding*, 159:3–18, June 2017.

[184] Y. Tian, D. Krishnan, and P. Isola. Contrastive representation distillation. *CoRR*, abs/1910.10699, January 2020.

[185] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, Las Condes, Chile, December 2015.

[186] G. Tsagkatakis, M. Jaber, and P. Tsakalides. Goal!! event detection in sports video. *Journal of Electronic Imaging*, 2017(16):15–20, January 2017.

[187] T. Tsunoda, Y. Komori, M. Matsugu, and T. Harada. Football action recognition using hierarchical LSTM. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 155–163, Honolulu, Hawaii, USA, July 2017.

[188] P. Tumas, A. Jonkus, and A. Serackis. Acceleration of HOG based pedestrian detection in FIR camera video stream. In *Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pages 1–4, Vilnius, Lithuania, April 2018.

[189] F. Turchini, L. Seidenari, L. Galteri, A. Ferracani, G. Becchi, and A. Del Bimbo. Flexible automatic football filming and summarization. In *ACM International Conference on Multimedia (ACM-MM) Workshops*, pages 108–114, Nice, France, October 2019.

[190] N. Ul Huda, B. Hansen, R. Gade, and T. Moeslund. The effect of a diverse dataset for transfer learning in thermal person detection. *Sensors*, 20(7):1–17, April 2020.

[191] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequièvre. A benchmark dataset for outdoor foreground/background extraction. In *Asian Conference on Computer Vision (ACCV)*, volume 7728 of *Lecture Notes in Computer Science*, pages 291–300. Springer, November 2012.

[192] B. Vanderplaetse and S. Dupont. Improved soccer action spotting using both audio and video streams. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), CVsports*, pages 3921–3931, Seattle, Washington, USA, June 2020.

[193] K. Vats, M. Fani, P. Walters, D. Clausi, and J. Zelek. Event detection in coarsely annotated sports videos via parallel multi receptive field 1D convolutions. *CoRR*, abs/2004.06172, April 2020.

[194] J. Vertens, A. Valada, and W. Burgard. SMSnet: Semantic motion segmentation using deep convolutional neural networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 582–589, Vancouver, BC, Canada, September 2017.

[195] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 734–741, Nice, France, October 2003.

[196] L. Wang and K.-J. Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *CoRR*, abs/2004.05937, April 2020.

[197] T. Wang, C. Chang, and Y. Wu. Template-based people detection using a single downward-viewing fisheye camera. In *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 719–723, Xiamen, China, November 2017.

[198] T. Wang and C. Liao. People detection in downward-viewing fisheye camera networks using fuzzy integral. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–5, New Orleans, Louisiana, USA, June 2019.

[199] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. CDnet 2014: An expanded change detection benchmark dataset. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 393–400, Columbus, Ohio, USA, June 2014.

[200] Y. Wang, Z. Luo, and P.-M. Jodoin. Interactive deep learning method for segmenting moving objects. *Pattern Recognition Letters*, 96:66–75, September 2017.

[201] K. Weiss, T. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–9, May 2016.

[202] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 90–97, Beijing, China, October 2005.

[203] T. Wu, S. Tang, R. Zhang, and Y. Zhang. CGNet: A light-weight context guided network for semantic segmentation. *CoRR*, abs/1811.08201, November 2018.

[204] J. Xie, B. Shuai, J.-F. Hu, J. Lin, and W.-S. Zheng. Improving fast segmentation with teacher-student learning. In *British Machine Vision Conference (BMVC)*, pages 1–13, Newcastle, United Kingdom, September 2018.

[205] M. Xu, N.C. Maddage, and C. Xu. Creating audio keywords for event detection in soccer video. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 2, pages 281–284, Baltimore, USA, July 2003.

[206] Y. Yang and D. Li. Robust player detection and tracking in broadcast soccer video based on enhanced particle filter. *Journal of Visual Communication and Image Representation*, 46:81–94, July 2017.

[207] Y. Yang, M. Xu, W. Wu, R. Zhang, and Y. Peng. 3D multiview basketball players detection and localization based on probabilistic occupancy. In *Digital Image Computing: Techniques and Applications*, pages 1–8, Canberra, Australia, December 2018.

[208] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, May 2018.

[209] J. Yim, D. Joo, J. Bae, and J. Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138, Honolulu, Hawaii, USA, July 2017.

[210] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *European Conference on Computer Vision (ECCV)*, volume 11217 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2018.

[211] H. Yu, S. Cheng, B. Ni, M. Wang, J. Zhang, and X. Yang. Fine-grained video captioning for sports narrative. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6006–6015, Salt Lake City, Utah, USA, June 2018.

[212] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *CoRR*, abs/1612.03928, June 2017.

[213] A. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3712–3722, Salt Lake City, Utah, USA, June 2018.

[214] H. Zawbaa, N. El-Bendary, A. Hassanien, and A. Abraham. SVM-based soccer video summarization system. In *Third World Congress on Nature and Biologically Inspired Computing (NaBIC)*, pages 7–11, Salamanca, Spain, October 2011.

[215] D. Zecha, M. Einfalt, and R. Lienhart. Refining joint locations for human pose tracking in sports videos. In *IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2524–2532, Long Beach, California, USA, June 2019.

[216] R. Zeng, W. Huang, M. Tan, Y. Rong, P. Zhao, J. Huang, and C. Gan. Graph convolutional networks for temporal action localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7093–7102, Seoul, Korea, October 2019.

[217] H. Zhang, B. Zhao, L. Tang, J. Li, and J. Li. Variational-based contour tracking in infrared imagery. In *International Congress on Image and Signal Processing*, pages 1–5, Tianjin, China, October 2009.

[218] L. Zhang, Y. Lu, G. Song, and H. Zheng. RC-CNN: Reverse connected convolutional neural network for accurate player detection. In *Pacific Rim International Conference on Artificial Intelligence (PRICAI): Trends in Artificial Intelligence*, pages 438–446, Nanjing, China, December 2018. Springer International Publishing.

[219] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. ICNet for real-time semantic segmentation on high-resolution images. In *European Conference on Computer Vision (ECCV)*, volume 11207 of *Lecture Notes in Computer Science*, pages 418–434, Munich, Germany, 2018.

[220] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Implementation of PSPNet. https://github.com/hszhao/PSPNet, 2016.

[221] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, Honolulu, Hawaii, USA, July 2017.

[222] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 74–95, Venice, Italy, October 2017.

[223] W.-B. Zheng, K.-F. Wang, and F.-Y. Wang. Background subtraction algorithm with Bayesian generative adversarial networks. *Acta Automatica Sinica*, 44(5):878–890, May 2018.

[224] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *CoRR*, abs/1608.05442, August 2016.

[225] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, Honolulu, Hawaii, USA, July 2017.

[226] L. Zhou, Y. Zhou, J. Corso, R. Socher, and C. Xiong. End-to-end dense video captioning with masked transformer. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8739–8748, Salt Lake City, Utah, USA, June 2018.