Localisation temporelle et suivi de l'action dans les vidéos de sport amateur

Axel Baldanza

▶ **To cite this version:**

Axel Baldanza. Localisation temporelle et suivi de l'action dans les vidéos de sport amateur. Autre [cs.OH]. Université de Bordeaux, 2023. Français. NNT : 2023BORD0233 . tel-04267901

HAL Id: tel-04267901
https://theses.hal.science/tel-04267901

Submitted on 2 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université de BORDEAUX

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE MATHÉMATIQUES ET INFORMATIQUE

MATHÉMATIQUES APPLIQUÉES ET CALCUL SCIENTIFIQUE

Par **Axel BALDANZA**

LOCALISATION TEMPORELLE ET SUIVI DE L'ACTION DANS LES VIDÉOS DE SPORT AMATEUR

Sous la direction de : **Jean-François AUJOL**
Co-directeur : **Yann TRAONMILIN**

Soutenue le 04/10/2023

Membres du jury :

M. AUJOL Jean-François	Professeur	Université de Bordeaux	Directeur de thèse
M. TRAONMILIN Yann	Chargé de recherche	Université de Bordeaux	Directeur de thèse
M. GOUSSEAU Yann	Professeur	Université Télécom ParisTech	Rapporteur
M. DENIS Loïc	Professeur	Université de Saint-Étienne	Rapporteur
Mme. PUSTELNIK Nelly	Directrice de Recherche	ENS Lyon	Présidente du jury
M. PAPADAKIS Nicolas	Directeur de Recherche	Université de Bordeaux	Examinateur
M. ALARY Francois	Directeur Technique	Rematch	Invité

Remerciements

Pour commencer, je tiens à remercier M. Jean-François AUJOL, Professeur à l'Université de Bordeaux et M. Yann TRAONMILIN, Chargé de recherche CNRS à l'Université de Bordeaux, les directeurs de ma thèse. Ils ont du lire un bon nombre de brouillons avant de pouvoir lire quelque chose qui s'apparente à un papier scientifique. Merci pour votre patience, pour vos conseils, et pour toute la sympathie dont vous avez fait preuve. Cette thèse n'aurait pas pu être sans vous.

Je voudrais ensuite remercier M. Yann GOUSSEAU, Professeur à Télécom ParisTech, et M. Loic DENIS, Professeur à l'Université de Saint-Étienne pour avoir accepté d'examiner cette thèse dans sa totalité, ainsi que les autres membres du jury de thèse pour le temps précieux qu'ils m'accordent, M. Nicolas PAPADAKIS, Directeur de Recherche CNRS à l'Université de Bordeaux et Mme. Nelly PUSTELNIK, Directrice de Recherche CNRS à l'ENS Lyon.

Je souhaite également exprimer mon immense gratitude à M. François ALARY, M. Pierre HUSSON, M. Franck SI-HASSEN ainsi qu'à toute la famille de Rematch, Maïky, Coco, Léo, Axel, Victor, Florent, Sonia, Florian, Ben, Yohan et les autres qui ont rendu ces années magiques. Merci pour votre implication dans ce projet, pour la liberté que vous m'avez laissée, merci pour les rires, pour votre ambition, et pour tout ce que vous apportez au sport amateur. À ce sujet, merci à tous ces gens mordus de ballons, quelle qu'en soit la forme. Sans vous, il n'y aurait pas d'actions loupées à 3 mètres des buts, pas de vidéos, pas de Rematch, et rien de tous ces longs remerciements.

Enfin, je tiens à remercier ma famille, mes frères, et surtout mes parents, Renaud et Martine, qui m'ont toujours encouragé à faire ce que j'aime et seulement ce que j'aime. Merci à mes amis, ceux du départ et ceux qui rejoignent le bateau au fur et à mesure. Merci pour les rires et pour l'influence que vous avez eue qui a fait de moi ce que je suis. Merci à tous les gens, peu importe qui ils sont, qui poussent les autres à essayer. Plus particulièrement à tous ces auteurs qui m'ont fait comprendre qu'il fallait se donner la peine de vivre quelque chose d'extraordinaire.

« *Tu sais, le silence ne fait que cinq bruits :*
Celui du regret, celui du "Pourquoi ne l'ai-je pas fait ?"
"Pourquoi ne l'ai-je pas dit ?", "Pourquoi vais-je taffer ?"
Celui du battement de cœur qui ne s'arrête pas car il n'a qu'une seule vie. »

William K. Mwamba.

Localisation temporelle et suivi de l'action dans les vidéos de sport amateur

Résumé : L'analyse des vidéos sportives est un domaine qui a récemment connu un attrait particulier en vision par ordinateur. Les objectifs sont multiples : améliorer les performances des athlètes, améliorer l'expérience visuelle des spectateurs, analyser les parties, faciliter l'acquisition ... Cependant, la majorité des travaux du domaine concernent le sport professionnel. Le sport amateur, qui concentre beaucoup moins de moyens, est largement moins étudié. Pourtant, ce contexte suscite de nombreuses nouvelles problématiques. Dans cette thèse, nous nous concentrons sur l'analyse de vidéos sportives capturées dans le contexte amateur. Plus précisément, nous traitons dans ce manuscrit deux problématiques précises : la localisation temporelle de l'action pour la découpe automatique des vidéos et le suivi vidéo pour la conception de caméras autonomes. Ainsi, la première partie de la thèse traite de la problématique de la localisation temporelle de l'action, aussi appelée proposition temporelle d'actions. Cette partie présente une méthode qui vise à la conception d'un algorithme de découpage automatique des vidéos en fonction de leur contenu. Ainsi, chaque vidéo traitée ne contient plus que l'action de jeu. Pour cela, nous traitons la problématique de proposition temporelle d'actions en proposant une méthode basée sur l'utilisation d'un réseau de neurones convolutif, qui permet de réduire le nombre de faux négatifs afin de limiter la possibilité d'endommager une séquence qui contient l'action de jeu. Dans les deux parties suivantes, nous traitons de la problématique de suivi vidéo. Il s'agit là de présenter une méthode qui vise à être embarquée dans une solution d'acquisition autonome et qui permet, à partir de l'image capturée par une caméra, de calculer ses déplacements nécessaires afin qu'elle suive l'action de jeu pour pouvoir la capter de manière autonome. Le contexte amateur suscitant des complexités spécifiques, nous présentons dans ces parties une nouvelle approche basée sur l'apprentissage automatique et l'apprentissage profond pour calculer ce suivi.
Mots-clés : Apprentissage profond, analyse vidéo, sport, localisation temporelle d'action, suivi vidéo, apprentissage automatique

---

Action temporal localization and tracking in amateur sport videos

Abstract: The analysis of sport videos is a domain that has recently come under particular interest in computer vision. The objectives are multiple: improving the performances of athletes, enhancing the visual experience of spectators, analyzing games, facilitating acquisition ... However, the majority of work in this area concerns professional sport. Amateur sport, where resources are much more limited, is less studied. Hence, this context gives rise to many new issues. In this thesis, we focus on the analysis of sport videos captured in the amateur context. More precisely, we address two specific problems in this manuscript: temporal localization of action for automatic video trimming, and video tracking for autonomous camera design. Thus, the first part of the thesis deals with the problem of temporal localization of action, also known as action temporal proposals. This part presents a method that aims to design an algorithm for automatically trimming videos according to their content. As a result, each processed video contains only the game action. To achieve this, we address the problem of temporal action proposals by proposing a method based on the use of a convolutional neural network, which reduces the number of false negatives in order to limit the possibility of damaging a sequence that contains the game action. In the next two sections, we address the problem of video tracking. This involves presenting a method that aims to be embedded in an autonomous acquisition solution and which, based on the image captured by a camera, calculates its necessary displacements so that it follows the game action in order to be able to capture it autonomously. As the amateur context gives rise to specific complexities, we present in these sections a new approach based on automatic learning and deep learning to calculate this tracking.
Keywords: Deep learning, video analysis, sport, action temporal localization, video tracking, machine learning

---

Unité de recherche
Institut de Mathématiques de Bordeaux (IMB) UMR 5251, 351 cours de la Libération, 33405 Talence, France.

Table des matières

1 Introduction **13**

1.1 Contexte et objectifs industriels 13

1.1.1 Contexte industriel général 13

1.1.2 Objectifs industriels 14

1.2 Contexte scientifique 15

1.2.1 Localisation temporelle d'actions 16

1.2.2 Contrôle autonome de caméra par suivi vidéo 18

1.3 Plan du manuscrit 19

1.4 Publications et contribution industrielle 20

1.4.1 Publications 20

1.4.2 Contribution industrielle 20

1.5 Notations générales 21

2 Découpage automatique des vidéos **23**

2.1 Introduction 23

2.1.1 Motivations 26

2.1.2 Notre méthode 27

2.1.3 Notation 27

2.2 État de l'art 27

2.2.1 Localisation temporelle d'actions 27

2.2.2 Classification d'images 31

2.2.3 Position de nos travaux et contribution 33

2.3 Analyse du contenu par descripteurs 34

2.3.1 Descripteurs colorimétriques 34

2.3.2 Étude des variations 37

2.4 Proposition/validation temporelle d'action 38

2.4.1 Première méthode : détecteur de joueurs 38

2.4.2 Deuxième méthode : AxionNet 40

2.5 Résultats 44

2.5.1 Métrique d'évaluation . . . 46
2.5.2 Évaluation . . . 47
2.6 Conclusion du chapitre . . . 49

3 Suivi d'action pour caméra autonome : modèle linéaire par morceaux 51
3.1 Introduction . . . 52
3.1.1 Motivations . . . 53
3.1.2 Notre méthode . . . 55
3.2 État de l'art . . . 55
3.2.1 Suivi d'objets . . . 56
3.2.2 Suivi d'objets dans les vidéos de sport . . . 60
3.2.3 Positionnement de notre méthode . . . 62
3.3 Flux optique et segmentation des plans . . . 63
3.3.1 Segmentation premier plan/arrière-plan . . . 64
3.3.2 Normalisation . . . 65
3.4 Modèle de prédiction linéaire par morceaux . . . 66
3.4.1 Modèle linéaire . . . 66
3.4.2 Modèle linéaire par morceaux . . . 67
3.4.3 Apprentissage et situations . . . 68
3.5 Bases de données et annotations . . . 69
3.5.1 Bases de données . . . 69
3.5.2 Annotation . . . 70
3.6 Résultats . . . 70
3.6.1 Métrique d'évaluation . . . 71
3.6.2 Évaluation . . . 72
3.7 Conclusion du chapitre . . . 76

4 Suivi d'action pour caméra autonome : réseau de neurones convolutif 79
4.1 Introduction . . . 79
4.1.1 Contexte . . . 79
4.1.2 Motivations . . . 80
4.2 Architecture CNN-3D pour le suivi d'action : NetCamMot . . . 81
4.2.1 Reformulation . . . 81
4.2.2 Détails de l'architecture . . . 82
4.3 Base de données de vidéos fisheye pour l'évaluation . . . 84
4.3.1 Vidéos . . . 84
4.3.2 Annotation . . . 85
4.4 Résultats . . . 85

4.4.1 Métrique d'évaluation 86
4.4.2 Comparaison avec le modèle linéaire par morceaux 86
4.4.3 Caméra virtuelle 91
4.5 Conclusion du chapitre 95

5 Conclusion **97**
5.1 Récapitulatif des contributions 98
5.1.1 Proposition temporelle d'actions 98
5.1.2 Suivi d'action vidéo 98
5.1.3 Publications 98
5.1.4 Intégrations 98
5.2 Perspectives et ouvertures 99

Table des figures

1.1 Exemples de tâches en vision par ordinateur. 15
1.2 Schéma illustrant le problème de proposition temporelle d'actions. 17
1.3 Schéma du pipeline de contrôle autonome de caméra PTZ. 18

2.1 Schéma illustrant notre problème. 24
2.2 Schéma d'une vidéo où l'action se situe au début. 24
2.3 Schéma d'une vidéo où l'action se situe à la fin. 25
2.4 Schéma d'une vidéo où l'action est présente dans toute la vidéo. 25
2.5 Schéma d'une vidéo endommagée. 25
2.6 Schéma illustrant le calcul du seuil IoU. 26
2.7 Schéma du principe de proposition temporelle d'actions par sliding window. 29
2.8 Schéma du principe d'ancres pour la proposition temporelle d'action. . . 30
2.9 Schéma du principe de proposition temporelle d'actions par prédiction du score d'action. 31
2.10 Schéma du principe d'apprentissage résiduel. 33
2.11 Architecture ResNet34. 33
2.12 Calcul de la détection de ciel D_1 sur une image d'action et une image de fond. 35
2.13 Calcul de la différence entre premier et dernier quart horizontal D_2 sur une image d'action et une image de fond. 35
2.14 Courbes obtenues à chaque étape du calcul. 36
2.15 Variations des descripteurs sur une vidéo. 37
2.16 Score du détecteur de joueurs en fonction du temps. 39
2.17 Résultats des détecteurs de personnes sur nos images. 40
2.18 Schéma explicatif de la méthode basée sur la détection de joueurs. 42
2.19 Schéma de l'architecture originale ResNet-18. 43
2.20 Schéma du pipeline de la deuxième méthode utilisant ResNet-18. 45

3.1 Schéma de l'objectif de notre méthode. 52

3.2 Schéma d'application de notre méthode sur les vidéos de nos bases de données. 53
3.3 Exemples d'images dans lesquelles on parvient à détecter le ballon. . . . 54
3.4 Exemples de fausses détections dans nos images. 54
3.5 Exemples d'images où le ballon n'est pas détecté. 55
3.6 Exemples d'images où le ballon est caché et donc pas détectable. 55
3.7 Schéma du système R-CNN pour la détection d'objets. 57
3.8 Schéma du fonctionnement d'une architecture siamoise. 59
3.9 Exemple de reconnaissance de numéros pour le suivi de joueurs par la méthode Joy19. 61
3.10 Exemples de segmentations réalisées grâce au flux optique. 65
3.11 Impact de la normalisation sur les segmentations. 67
3.12 Schéma du pipeline du modèle de prédiction linéaire par morceaux. . . . 68
3.13 Exemples des résultats du suivi sur les vidéos n°2, 7, 14 et 15 de la base de données. 73

4.1 Schéma du pipeline de test sur les vidéos capturées avec une caméra fisheye. 81
4.2 Schéma de l'architecture du réseau NetCamMot. 83
4.3 Pipeline détaillé de la méthode NetCamMot. 84
4.4 Exemples de portions d'images issues de vidéos inutilisables. 85
4.5 Évaluation de la précision du suivi dans les images fisheye. 87
4.6 Exemples de vidéos bien suivies par NetCamMot. 91
4.7 Courbes des résultats obtenus par NetCamMot sur une vidéo de basket-ball. 93
4.8 Courbes des résultats obtenus par NetCamMot sur une vidéo de handball. 94

Liste des tableaux

2.1 Résultats des modèles sur les vidéos non-bornées. 48
2.2 Nombre de vidéos endommagées sur toute la base de données en fonction de la méthode de localisation utilisée. 48
2.3 Pourcentage de vidéos bien découpées en fonction de la méthode de localisation utilisée sur toute la base de données. 49

3.1 Résultats des architectures pour la détection du ballon de la littérature sur nos images. 63
3.2 Résultats préliminaires du modèle sur la base de données des vidéos courtes de basket-ball. 72
3.3 Résultats du modèle linéaire par morceaux sur les bases de données des vidéos courtes de basket-ball et handball. 75
3.4 Résultats du modèle linéaire par morceaux sur les bases de données des vidéos longues de basket-ball. 76

4.1 Architecture du réseau 83
4.2 Résultats des différents modèles sur la base de données des vidéos courtes de basket-ball. 88
4.3 Résultats des différents modèles sur la base de données des vidéos courtes de handball. 89
4.4 Résultats des modèles NetCamMot et PEN sur la base de données des vidéos longues de basket-ball. 90
4.5 Résultats du modèle NetCamMot sur la base de données des vidéos fisheye. 92

Chapitre 1

Introduction

Sommaire

1.1 Contexte et objectifs industriels 13
1.1.1 Contexte industriel général 13
1.1.2 Objectifs industriels 14
1.2 Contexte scientifique 15
1.2.1 Localisation temporelle d'actions 16
1.2.2 Contrôle autonome de caméra par suivi vidéo 18
1.3 Plan du manuscrit 19
1.4 Publications et contribution industrielle 20
1.4.1 Publications 20
1.4.2 Contribution industrielle 20
1.5 Notations générales 21

1.1 Contexte et objectifs industriels

1.1.1 Contexte industriel général

Dans l'industrie, le traitement et l'analyse des vidéos sportives est un domaine largement étudié ces dernières années. Différents types de structures concentrent des efforts sur le sujet. Les productions télévisuelles développent des caméras de plus en plus performantes pour diffuser des images d'une très haute qualité et proposer des angles de vues immersifs et originaux. Elles développent également des algorithmes d'analyse et de compréhension vidéo afin de pouvoir proposer des statistiques détaillées des faits de jeu dans des délais très rapides. Les clubs et structures sportives professionnels font eux

appels à des entreprises pour développer des outils d'analyses vidéos qui visent à améliorer leurs performances (étude des performances des joueurs, analyse des déplacements, futurs adversaires ...).

Dans le sport professionnel, où les structures peuvent allouer des budgets conséquents, les outils peuvent être équipés de matériels onéreux (caméra très haute définition, caméras multiples, ordinateurs de calcul performants, ...) qui permettent d'obtenir des images de très bonne qualité et d'utiliser des opérateurs mathématiques avec des coûts computationnels élevés.

Dans le sport amateur, les moyens sont limités et les données moins disponibles, faisant apparaître de nouvelles problématiques. Lors des développements de produits matériels, les composants utilisés doivent être moins coûteux pour favoriser leur distribution. Les caméras moins performantes rendent le traitement et l'analyse des images plus difficiles. L'utilisation de processeurs moins puissants limite la complexité des opérateurs que l'on peut utiliser. Les données disponibles dans le sport amateur sont plus rares, et les images disponibles sont variables (point de vue, environnement, qualité), car souvent capturées par des personnes bénévoles avec des lentilles hétérogènes (téléphones, caméscopes, ...). Les méthodes utilisées dans le sport amateur doivent répondre à ces problématiques et traiter ces variations. Elles sont donc différentes de celles utilisées dans le sport professionnel.

Cette thèse s'inscrit dans le cadre d'un contrat CIFRE entre Rematch [1] et l'Institut de Mathématiques de Bordeaux (IMB).

1.1.2 Objectifs industriels

Dans ce contexte, nos travaux s'articulent autour de deux objectifs. Le premier vise à concevoir un algorithme de découpe automatique des vidéos de sport amateur en fonction de leur contenu. Les vidéos de nos bases de données sont capturées par des utilisateurs qui ne sont pas systématiquement concentrés sur l'action du match. Un grand nombre de vidéos contiennent des sous-séquences sans action de jeu qui nécessitent d'être supprimées. Un des objectifs de cette thèse est donc la conception d'un algorithme d'analyse vidéo qui permet de renvoyer les bornes temporelles du début et de fin des séquences d'action de jeu dans les vidéos afin de pouvoir supprimer les parties sans action. Ce travail a pour but de nous permettre le traitement des vidéos des bases de données de la plateforme Rematch pour améliorer la qualité de son contenu ainsi que pour pré-traiter les vidéos pour les chapitres suivants. En ne conservant que les séquences d'action, nous nous assurons d'un premier filtrage des images que nous ajouterons aux bases de données utilisées plus

1. Plateforme Rematch : www.rematch.tv

tard. Le deuxième objectif de ces travaux est la conception d'une caméra intelligente, capable de capturer le sport amateur de façon autonome. On traitera dans cette thèse de la conception d'une méthode qui vise à être embarquée dans une solution d'acquisition et qui permet de suivre l'action pour diriger la caméra vers la zone de l'espace où elle se situe.

1.2 Contexte scientifique

Le traitement des images et des vidéos, appelé **vision par ordinateur**, est un domaine des mathématiques appliquées qui provient de la volonté des humains à automatiser la compréhension visuelle et à améliorer la capture des images. Les domaines d'applications de la vision par ordinateur sont nombreux. Elle est notamment utilisée dans l'imagerie médicale (Suzuki, 2017), la vidéo-surveillance (Ibrahim, 2016) ou pour la navigation et la conception de véhicules autonomes (Janai *et al.*, 2020). Pour parvenir à imiter la compréhension visuelle humaine de la meilleure des façons, la vision par ordinateur se décline en de nombreuses tâches différentes telles que la classification d'image (Wang *et al.*, 2019), la détection d'objets (Zou *et al.*, 2023), le suivi d'objets (Luo *et al.*, 2021), la segmentation (Pustelnik *et al.*, 2016), l'inpainting (Newson *et al.*, 2014), ...

Classification d'image | Détection d'objets | Segmentation

FIGURE 1.1 – **Exemples de tâches en vision par ordinateur.**

Ces dernières années, l'apprentissage automatique (*Machine Learning*) et l'apprentissage profond (*Deep Learning*) ont connu des avancées significatives dans le domaine de la vision par ordinateur. Ces avancées ont permis des améliorations considérables dans les tâches que nous venons de citer. Notamment, les réseaux de neurones profonds ont été largement utilisés pour l'apprentissage automatique, offrant des performances supérieures à celles des méthodes traditionnelles. Shrestha et Mahmood (2019) présente une revue détaillée de l'apprentissage profond. Ce papier permet de comprendre son fonctionnement et retrace l'historique technique du domaine.

Pour répondre aux problématiques de cette thèse, nous allons nous concentrer principalement sur deux aspects de la vision par ordinateur : la **localisation temporelle d'ac-**

tions (*Action Temporal Localization*), et **le suivi vidéo** (*Video Tracking*).

1.2.1 Localisation temporelle d'actions

La localisation temporelle d'actions dans les vidéos est un sujet de la vision par ordinateur qui appartient au domaine de la compréhension d'actions. C'est un domaine vaste qui regroupe plusieurs objectifs : la **reconnaissance d'action** (*Action Recognition*) qui consiste à classifier de manière binaire une vidéo en fonction de l'activité présente dans son contenu. La **proposition temporelle d'actions** (*Temporal Action Proposal*) qui consiste à proposer des bornes de début et de fin des séquences d'action dans la vidéo. La **classification d'action** (*Action Classification*), qui consiste à classifier l'action en fonction de son contenu. La **localisation spatiale de l'action** (*Spatial Action Localization*) qui consiste à trouver l'endroit de l'image où a lieu l'action. Certaines de ces tâches peuvent être regroupées dans une même méthode. Par exemple, certains travaux présentent des méthodes de **détection d'actions** (*Action Detection*) qui consistent à trouver les bornes temporelles d'actions dans la vidéo et à classifier leur contenu.

Jiang *et al.* (2014) est un des travaux majeurs de ce domaine. Il définit la détection temporelle d'actions comme ceci :

Definition 1.2.1. La **détection temporelle d'actions** (*Temporal Action Detection*) vise à détecter tous les instants de début et de fin et classifier les catégories d'actions correspondantes à partir d'une vidéo non découpée.

Dans cette thèse, nous laissons de côté les thématiques de classification et de localisation spatiale pour nous concentrer sur la caractéristique temporelle : la **proposition temporelle d'actions.**

Proposition temporelle d'actions

La proposition temporelle d'actions est une tâche qui prend en entrée une vidéo non-bornée et consiste à échantillonner des portions temporelles de vidéo qui correspondent à l'action.

Definition 1.2.2. Une vidéo **non-bornée** ou non-découpée (*untrimmed video*) est une vidéo dans laquelle l'action n'est pas présente dans toute la vidéo. Les parties ne contenant pas l'action sont appelées "Fond" (*background*).

Cette tâche revient donc à prédire les bornes de début et de fin des actions dans la vidéo. Les sous-séquences sont alors binairement classifiées comme "Action" ou "Fond". La Figure 1.2 schématise le problème.

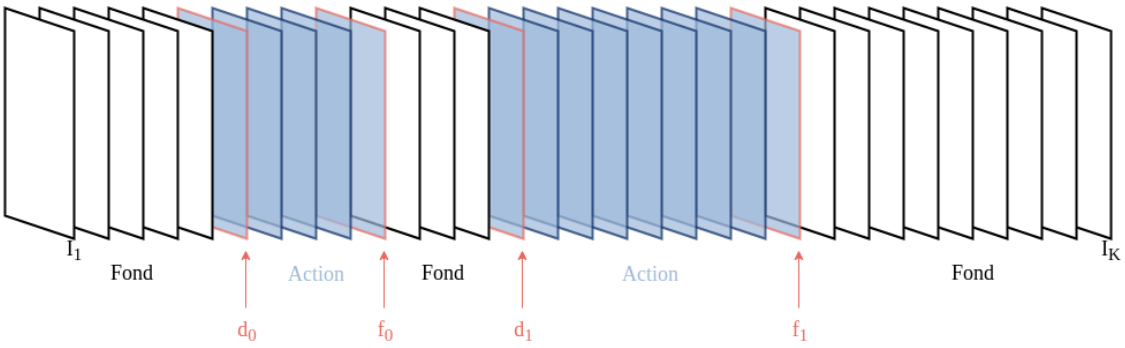


FIGURE 1.2 – **Schéma illustrant le problème de proposition temporelle d'actions.** Dans ce problème, on cherche à déterminer les bornes de début et de fin des actions ((d_0, f_0) et (d_1, f_1) dans le schéma).

Précision

Dans la littérature, l'évaluation des propositions temporelles d'actions se fait principalement sur les bases de données présentées par ActivityNet (Fabian Caba Heilbron et Niebles, 2015) et THUMOS14 (Jiang *et al.*, 2014). Elles permettent également d'évaluer la classification d'action. Pour évaluer les performances d'un modèle dans cette tâche, ces études utilisent le **mAP** (*mean Average Precision*), que nous présenterons plus tard. Cette métrique est calculée selon un seuil de précision appelé **tIoU** (*temporal Intersection over Union*) qui correspond au seuil minimum de la zone de recouvrement entre la portion d'action prédite et la portion d'action réelle par rapport à la zone couverte par l'union des deux. Ainsi, plus ce seuil est proche de 1, plus la zone prédite doit être proche de la zone réelle pour que la prédiction soit considérée comme correcte, résultat que l'on appelle vrai positif (*True Positive*). On parle alors de **mAP@tIoU** pour présenter les résultats d'un modèle à un certain niveau de précision requis. Ainsi, on constate dans la littérature, que beaucoup de méthodes obtiennent des résultats performants à des tIoU en dessous de 0.7, mais que les performances chutent fortement lorsque ce seuil se rapproche de 1. Ce constat provient de plusieurs phénomènes. Premièrement, les positions précises de début et de fin d'une action peuvent être subjectives et changent en fonction de la personne qui les annote. Ensuite, dans certaines méthodes, l'intérêt principal visé se situe dans la classification du type d'action plus que dans la précision de sa localisation temporelle. Enfin, la détermination précise des bornes peut s'avérer une tâche très compliquée lorsque les régions de "fond" sont contextuellement proches des régions d'actions. Dans cette thèse, nous traiterons de cette question de précision et de comment limiter son impact négatif pour la découpe automatique de vidéos.

1.2.2 Contrôle autonome de caméra par suivi vidéo

La problématique de contrôle autonome de caméra est un vaste sujet qui se décline en de nombreuses applications très différentes. Ce sujet regroupe le contrôle d'**une caméra seule** (*single camera autonomous control*), le contrôle de **caméras multiples** (*multi-cameras autonomous control*) ainsi que tous les systèmes de **navigation autonome** (*autonomous navigation*) de véhicules ou robots qui visent à commander des déplacements autonomes à partir des images acquises par une ou plusieurs caméras embarquées. Dans cette thèse on traitera de la problématique du contrôle de caméra unique que l'on appelle caméra PTZ (*Pan-Tilt-Zoom*).

Definition 1.2.3. Une caméra PTZ (*Pan-Tilt-Zoom*) est un type de caméra vidéo capable d'être pilotée par elle-même. Le terme "Pan" désigne le mouvement horizontal de l'objectif, le terme "Tilt" désigne le mouvement vertical de l'objectif et le terme "Zoom" désigne le réglage de la longueur focale de l'objectif. Ces termes regroupent les différents axes de pilotage possibles de la caméra.

La problématique de contrôle autonome de caméra PTZ est née entre la fin des années 1990 et le début des années 2000 avec l'arrivée de ce type de caméra. Bagdanov *et al.* (2006) et Kumar *et al.* (2009) présentent en 2006 et 2009 des papiers qui développent des pipelines complets permettant de diriger une caméra PTZ à partir de l'analyse du contenu capturé par cette même caméra. Notamment, Bagdanov *et al.* (2006) détecte et suit les visages dans les images pour diriger la caméra. Dans ces papiers, les pipelines de contrôle autonome sont divisés en deux étages : un module de suivi et un module de contrôle. Ce pipeline est schématisé dans la Figure 1.3. Le module de suivi consiste à analyser l'image pour déterminer l'objet où la zone à suivre et le module de contrôle vise à diriger la caméra vers cette zone. C'est principalement la partie suivi qui nous intéresse.

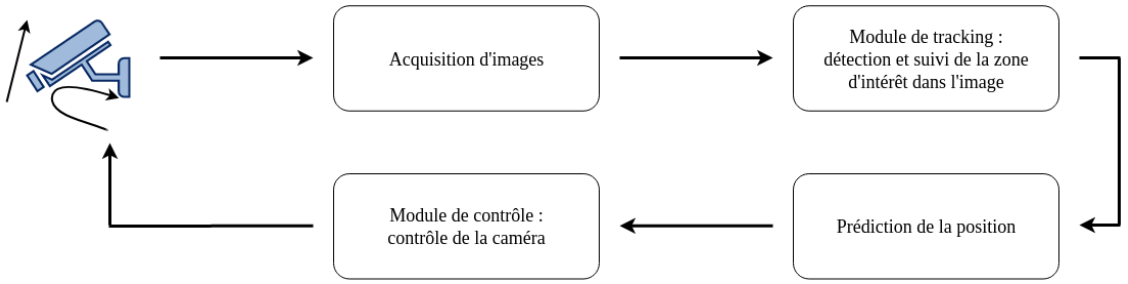


FIGURE 1.3 – **Schéma du pipeline de contrôle autonome de caméra PTZ.**

Suivi vidéo

Le suivi vidéo (*video tracking*) souvent associé au suivi d'objets (*object tracking*) est un domaine vastement étudié en vision par ordinateur. Il consiste à détecter et à suivre

un, plusieurs objets ou une zone d'intérêt tout au long de la vidéo. Pour ça, la plupart des méthodes utilisent le suivi par détection (*tracking-by-detection*) qui consiste à détecter le ou les objets dans toutes les images et à relier temporellement ces détections. Une des principales difficultés de cette tâche est de traiter les problèmes d'occlusion : continuer de suivre un objet lorsqu'il est caché par quelque chose et n'est donc pas détectable.

On distingue dans la littérature deux grands types de suivi : le **suivi d'objet unique** (*Single Object Tracking*) et le **suivi d'objets multiples** (*Multiple Object Tracking*). Le suivi d'objets multiples est une tâche beaucoup plus complexe car elle induit de reconnaître et de lier les objets entre eux pour pouvoir les suivre.

Contrainte temps réel

Un aspect du suivi mis en avant dans les thématiques de contrôle autonome est la contrainte temps réel. Les méthodes pour les systèmes d'acquisition embarqués doivent être calculées en même temps que l'action a lieu. Ce type de méthodes s'appelle les méthodes *online*. Elles s'opposent au traitement des vidéos après leur capture que l'on appelle méthodes *offline*. Prenons une cadence classique d'enregistrement qui se situe autour de 30 images par seconde (fps). En théorie, il faut que les algorithmes de suivi en temps réel puissent être calculés 30 fois par seconde. En réalité, il n'est pas toujours obligatoire de calculer le suivi sur chaque image. Pour qu'une méthode soit dite en temps réel, il faut que le temps de calcul de la méthode sur une image soit inférieur au ratio entre une seconde et le nombre d'images nécessaires pour effectuer le suivi dans une seconde.

1.3 Plan du manuscrit

Après cette introduction, ce manuscrit s'articule en 4 chapitres. Le Chapitre 2 traite de la localisation temporelle de l'action pour la découpe automatique des vidéos. Il développe dans un premier temps un état de l'art détaillé du sujet avant de présenter les travaux réalisés et les résultats obtenus. Dans cette partie, nous présentons une méthode de proposition/validation des bornes de début et de fin de l'action afin de réduire le nombre de faux positifs (**AxionNet**). Cette méthode a pour but de répondre au problème de découpage automatique de vidéos en limitant le nombre de **vidéos endommagées**.

Les Chapitres 3 et 4 sont dédiés au suivi d'action. Ces parties visent à présenter un algorithme qui, une fois embarqué, permet la capture automatique d'actions dans le sport amateur. Nous présentons dans ces chapitres des méthodes qui réalisent le suivi de l'action en prédisant les mouvements de la caméra nécessaires au suivi du centre de l'action. Ces méthodes basent cette prédiction sur les mouvements des joueurs dans l'image.

Le Chapitre 3 est construit comme le précédent. Il présente dans un premier temps un état de l'art détaillé du sujet, et développe ensuite la méthode et les résultats. Ce chapitre présente une méthode d'apprentissage linéaire par morceaux pour répondre au problème.

Dans le Chapitre 4, nous présentons **NetCamMot**, une méthode qui reprend les principes du chapitre précédent et utilise un réseau de neurones convolutif pour calculer la prédiction. Ce chapitre présente l'architecture utilisée et les résultats obtenus. On compare notamment dans cette partie les résultats obtenus avec ceux du chapitre précédent pour mettre en avant l'utilisation de notre nouvelle architecture.

Ce manuscrit est conclu par le Chapitre 5. Dans ce chapitre, nous résumons les différentes contributions présentées par nos travaux et ouvrons la discussion sur les perspectives et les différentes limites qui découlent de nos travaux.

1.4 Publications et contribution industrielle

1.4.1 Publications

Ces travaux ont donné lieu à des publications scientifiques nationales et internationales :

— **A. Baldanza**, J.-F. Aujol, Y. Traonmilin, and F. Alary. *Découpage automatique de vidéos de sport amateur par détection de personnes et analyse de contenu colorimétrique.* ORASIS 2021.

— **A. Baldanza**, J.-F. Aujol, Y. Traonmilin, and F. Alary. *Piecewise linear prediction model for action tracking in sports.* Proceedings of the 30th European Signal Processing Conference (EUSIPCO 2022), 2022.

— **A. Baldanza**, J.-F. Aujol, Y. Traonmilin, and F. Alary. *Real-time multi-sport action tracking with convolutional neural networks*, preprint HAL, 2022.

1.4.2 Contribution industrielle

Ces travaux ont également permis la mise en production d'algorithmes dans le pipeline d'encodage vidéo de l'application Rematch. La méthode de découpage automatique présentée dans le Chapitre 2 est utilisée pour le traitement automatique des vidéos sur la plateforme. Chaque vidéo enregistrée sur l'application est désormais traitée par notre méthode pour améliorer la pertinence et la qualité du contenu proposé aux utilisateurs. Les travaux qui concernent le suivi vidéo jouent un rôle prédominant dans la conception d'une solution d'acquisition autonome pour le sport amateur qui devrait voir le jour prochainement.

1.5 Notations générales

Ces notations sont utilisées dans toute la thèse.

Soit I une image de taille $N \times M$ pixels, on considérera I comme la fonction définie par :

$$\begin{array}{rcccc} I & : & [\![1;N]\!] \times [\![1;M]\!] & \rightarrow & \mathbb{R}^3 \\ & & (x,y) & \mapsto & I(x,y), \end{array} \tag{1.1}$$

qui à un pixel (x, y) associe ses valeurs **RGB**.

Dans la suite, on considère une vidéo comme composée de K images. I_j est l'image de taille $N \times M$ définie comme ci-dessus qui représente la j-ème image de la vidéo, $j \in [\![1;K]\!]$. Les indices R, G et B : I^R, I^G et I^B permettent dans le cas où les calculs sont faits sur un seul canal couleur et non toute l'image, de préciser de quel canal il s'agit.

Chapitre 2

Découpage automatique des vidéos

Sommaire

2.1 Introduction 23
2.1.1 Motivations 26
2.1.2 Notre méthode 27
2.1.3 Notation 27
2.2 État de l'art 27
2.2.1 Localisation temporelle d'actions 27
2.2.2 Classification d'images 31
2.2.3 Position de nos travaux et contribution 33
2.3 Analyse du contenu par descripteurs 34
2.3.1 Descripteurs colorimétriques 34
2.3.2 Étude des variations 37
2.4 Proposition/validation temporelle d'action 38
2.4.1 Première méthode : détecteur de joueurs 38
2.4.2 Deuxième méthode : AxionNet 40
2.5 Résultats 44
2.5.1 Métrique d'évaluation 46
2.5.2 Évaluation 47
2.6 Conclusion du chapitre 49

2.1 Introduction

Ce chapitre est dédié à la conception d'une méthode de localisation temporelle d'actions pour le découpage automatique des vidéos. Comme expliqué dans l'introduction,

les modalités d'acquisition des vidéos de sport amateur engendrent la présence de sous-séquences sans action de jeu, appelées séquences de "Fond". Dans cette étude, une séquence est définie comme "Action" si elle contient le terrain de jeu et les joueurs. Elle est classifiée comme "Fond" si elle contient autre chose. On cherche à prédire les bornes de début et de fin de ces séquences pour pouvoir supprimer les séquences de fond. La méthode prend en entrée une vidéo et retourne les bornes temporelles de l'action dans la vidéo. La base de données qui constitue cette étude est composée de vidéos où la taille et la position des séquences d'action sont variables. Les vidéos de cette base sont courtes (15 secondes maximum). Par conséquent, on suppose qu'une vidéo ne contient qu'une seule action à localiser. Soit V une vidéo composée de K images. On cherche à déterminer $(J_1, J_2) \in [\![1; K]\!]^2$ respectivement les bornes de début et de fin de l'action dans la vidéo. La méthode peut facilement être adaptée pour prédire plus de deux bornes.

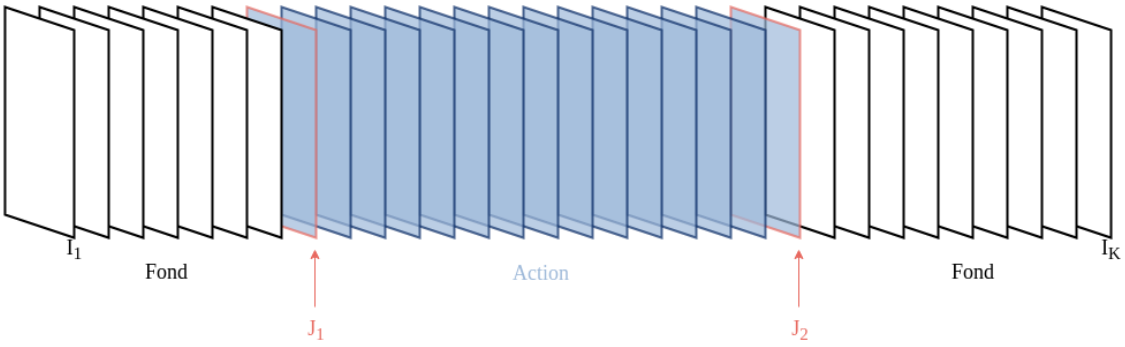


FIGURE 2.1 – **Schéma illustrant le problème.** Ici, on cherche à déterminer les bornes J_1 et J_2 de début et de fin de l'action dans la vidéo.

La position et la longueur des actions ne répondent à aucune règle spécifique. Elle peut se situer au début de la vidéo (Figure 2.2), au milieu (Figure 2.1), ou à la fin (Figure2.3).

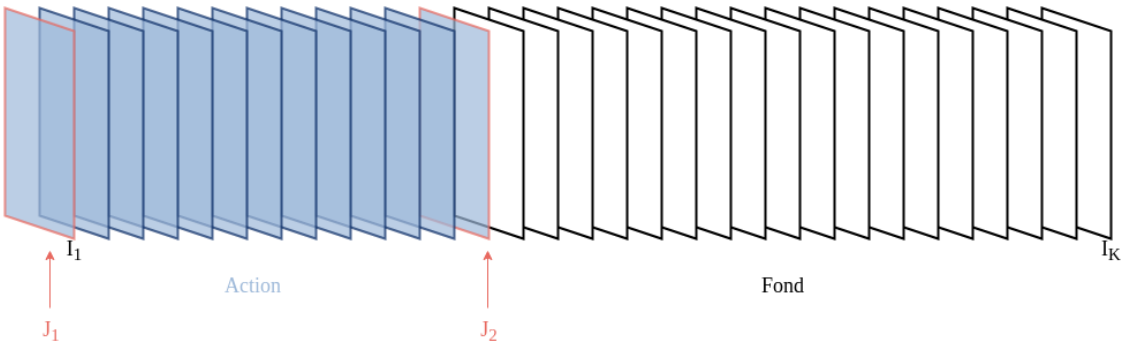


FIGURE 2.2 – **Schéma d'une vidéo où l'action se situe au début.**

Une des particularités de cette méthode est qu'elle doit également fonctionner sur des vidéos qui ne contiennent que l'action. Dans ce cas, elle doit renvoyer comme bornes le début et la fin de la vidéo comme dans la Figure 2.4.

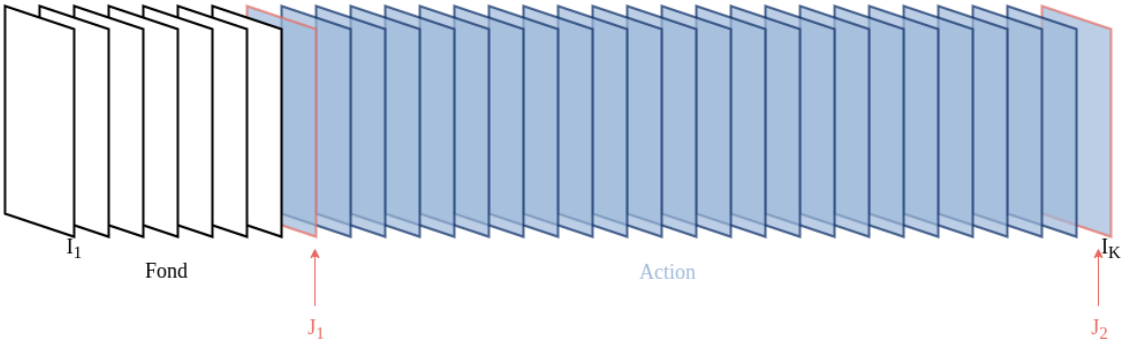


FIGURE 2.3 – **Schéma d'une vidéo où l'action se situe à la fin.**

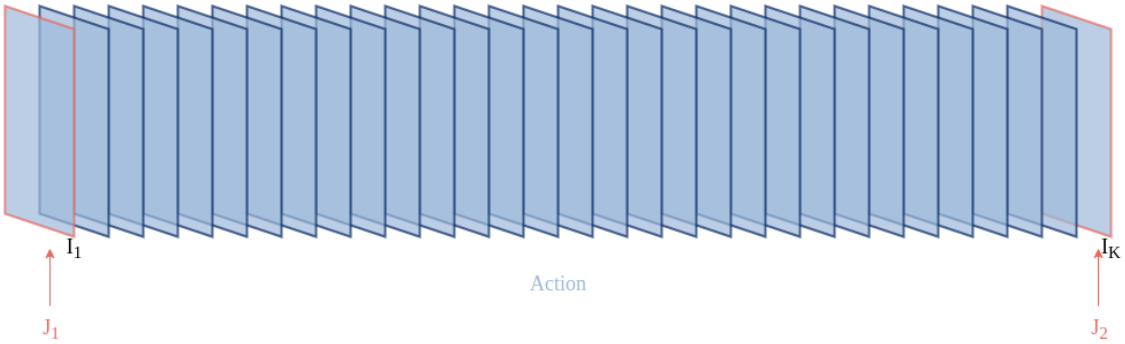


FIGURE 2.4 – **Schéma d'une vidéo où l'action est présente dans toute la vidéo.**

Il est important de remarquer que lorsqu'on définit une méthode de localisation temporelle pour concevoir un algorithme de découpage automatique, une mauvaise localisation peut endommager une vidéo. On définit une **vidéo endommagée** comme une vidéo où la sous-séquence entre une borne prédite et la vérité terrain contient une séquence d'action (à un seuil près). Ce type de cas est illustré dans la Figure 2.5

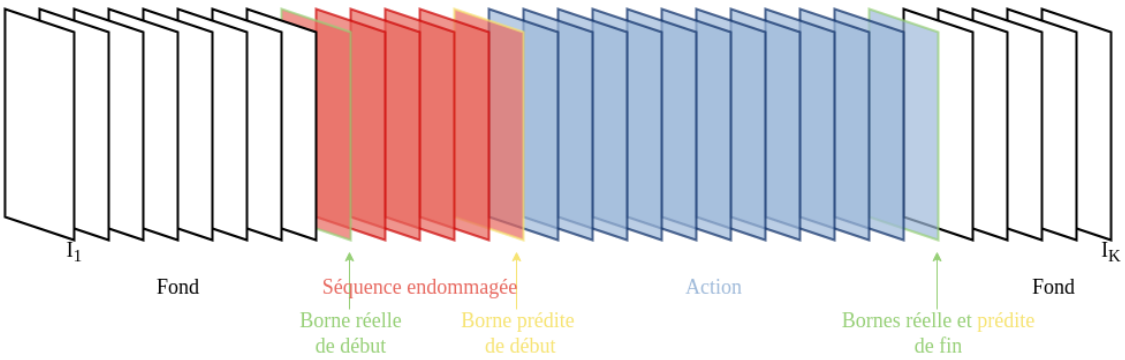


FIGURE 2.5 – **Schéma d'une vidéo endommagée.**

Dans notre étude, l'objectif est de limiter ce phénomène pour ne pas perdre les actions de jeu.

2.1.1 Motivations

Aujourd'hui, les méthodes de localisation temporelle les plus performantes obtiennent d'excellents résultats selon les métriques d'évaluation classiques définies pour des études telles que ActivityNet (Fabian Caba Heilbron et Niebles, 2015) et THUMOS14 (Jiang *et al.*, 2014). Cependant, lorsqu'on regarde les résultats de ces modèles, on constate que les performances diminuent fortement pour des seuils de tIoU (temporal Intersection over Union) élevés. Ce seuil est introduit par Jaccard dans Jaccard (1901) comme mesure de similarité entre deux échantillons. Elle est aujourd'hui très utilisée en vision par ordinateur pour évaluer les méthodes de détection. Elle consiste à comparer la zone de recouvrement entre la prédiction et la vérité terrain avec l'union des deux, comme le montre la Figure 2.6.

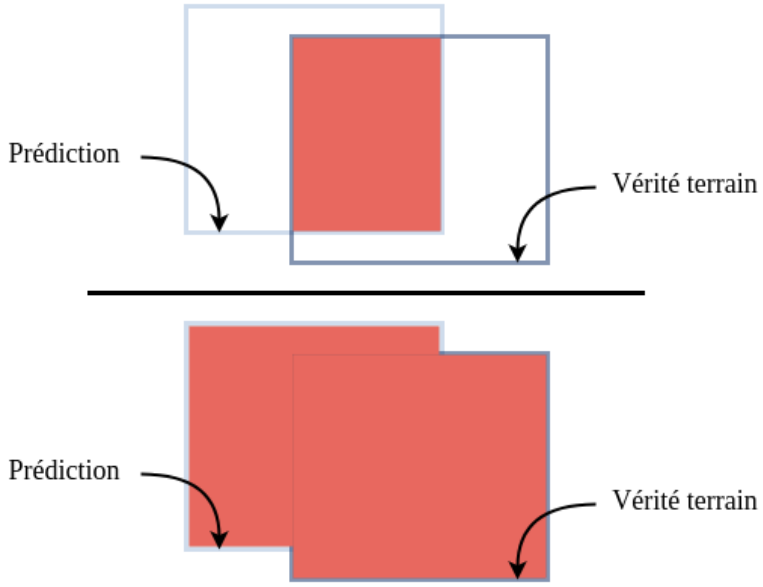


FIGURE 2.6 – **Schéma illustrant le calcul du seuil tIoU.** Ce seuil permet de déterminer la valeur minimum de superposition entre la prédiction et la vérité terrain pour qu'une prédiction soit considérée comme correcte.

Pour la détection d'objets, on parle d'Intersection over Union (IoU), qui a été décliné en temporal Intersection over Union (tIoU) pour la détection d'actions. Cet indice se calcule de la façon suivante :

$$tIoU([\hat{J}_1, \hat{J}_2], [J_1, J_2]) = \frac{|[\hat{J}_1, \hat{J}_2] \cap [J_1, J_2]|}{|[\hat{J}_1, \hat{J}_2] \cup [J_1, J_2]|} \tag{2.1}$$

Ainsi, le seuil tIoU permet de définir le niveau de précision requis pour qu'une prédiction soit considérée comme un vrai positif. Comme expliqué plus haut, des prédictions peu précises peuvent engendrer la découpe de séquences d'actions. La baisse des perfomances des modèles dans la littérature pour des tIoU élevés justifie l'importance de définir

une méthode qui vise à la localisation temporelle d'actions en limitant les faux négatifs.

2.1.2 Notre méthode

Pour cela, nous allons présenter une méthode qui prédit une région d'action via un processus de proposition/validation des bornes. Pour calculer cette validation, notre méthode consiste à définir des descripteurs qui permettent de discriminer les séquences à conserver de celles à supprimer et d'étudier leurs variations au cours du temps pour les comparer aux bornes de localisation proposées. Ce travail a été divisé en deux parties. Dans un premier temps, nous avons construit une méthode basée sur l'utilisation d'opérateurs non appris sur nos données. Cette méthode base la proposition des bornes sur l'étude des variations d'un détecteur de joueurs dans les images. Elle calcule la validation des propositions à partir de variations des descripteurs définis pour discriminer les séquences d'action des séquences de fond. Cette méthode est ensuite utilisée pour l'annotation semi-automatique d'une base de données servant à l'apprentissage de réseaux de neurones dédiés à la classification des images en fonction de leur contenu. Ce classificateur est utilisé dans une nouvelle méthode pour la proposition des bornes.

2.1.3 Notation

Cette notation est utilisée dans tout le chapitre. On définit D

$$\begin{array}{rcccc} D & : & [\![1;K]\!] & \rightarrow & [0;1] \\ & & j & \mapsto & D(j) \end{array} \tag{2.2}$$

une fonction qui associe à un temps j de la vidéo une valeur entre 0 et 1 calculée à partir de la j-ème image de la vidéo. L'étude de cette fonction au cours du temps a pour but d'indiquer les variations entre les séquences d'actions et de fond. Par exemple, un descripteur D qui renvoie 1 sur une séquence d'action et 0 sur une séquence de fond permettra de déduire précisément le moment de variation entre les deux sous-séquences.

2.2 État de l'art

2.2.1 Localisation temporelle d'actions

Les différents principes de la localisation temporelle d'actions sont présentés dans l'introduction de cette thèse. Dans la littérature, la localisation d'action est un sujet très étudié et essentiel de la vision par ordinateur.

Bases de données

Il existe plusieurs dizaines de bases de données qui permettent d'entraîner et d'évaluer les méthodes de compréhension d'action, mais la plupart des méthodes qui se concentrent sur la détection d'actions sont évaluées sur les deux principales : THUMOS14 (Jiang *et al.*, 2014) et ActivityNet (Fabian Caba Heilbron et Niebles, 2015). Ces bases sont constituées de vidéos non-bornées issues de la plateforme YouTube et permettent d'évaluer les tâches de proposition temporelle d'actions ainsi que de détection (proposition temporelle et classification de son contenu). La base de données Kinetics, présentée dans Carreira et Zisserman (2017) est également beaucoup utilisée, mais elle concerne seulement la classification des actions.

Comme il est expliqué dans l'introduction, nous nous concentrons ici sur la thématique de proposition temporelle d'actions et laissons de côté la classification du contenu. Certaines méthodes de détection d'action, utilisent des méthodes dites "*one stage*" qui prédisent la localisation et classifient l'action en même temps, mais la plupart des méthodes fonctionnent en deux temps (*two stages*) : proposition puis classification. En se concentrant sur la problématique de proposition temporelle d'actions, on distingue deux grandes familles de méthodes. Les méthodes basées sur les principes d'ancres (*anchors*) ou de fenêtres glissantes (*sliding-windows*), appelées méthodes descendantes (*top-bottom*), et les méthodes qui basent la proposition sur la prédiction d'un score d'action (*Actionness Score*) image par image, appelées méthodes ascendantes (*bottom-up*).

Ancres et sliding-window

Oneata *et al.* (2013); Shou *et al.* (2016) présentent des méthodes basées sur les fenêtres glissantes. Ces méthodes consistent à diviser une séquence vidéo en plusieurs segments et à appliquer une classification binaire (Action ou Fond) sur chacun de ces segments. Ainsi, une action est détectée à partir d'un ou plusieurs segments consécutifs contenant l'action. La Figure 2.7 présente schématiquement ce principe. Dans ce schéma, les segments ont une taille unique. Dans beaucoup d'applications, on génère des segments multi-échelles pour permettre au modèle plus de précision, on peut aussi parler d'échelle pyramidale.

Suivant un principe similaire, les méthodes basées sur les ancres (*anchors-based*) reprennent le principe d'ancre dans la détection d'objets. Ces méthodes consistent à définir des ancres qui couvrent l'ensemble de la vidéo. Chaque ancre est associée à une position temporelle et une taille spécifique, et représente une hypothèse de localisation possible pour une action dans la séquence vidéo. Ces ancres sont ensuite utilisées pour prédire la présence d'une action à un certain moment de la séquence vidéo. La Figure 2.8 schématise ce principe.

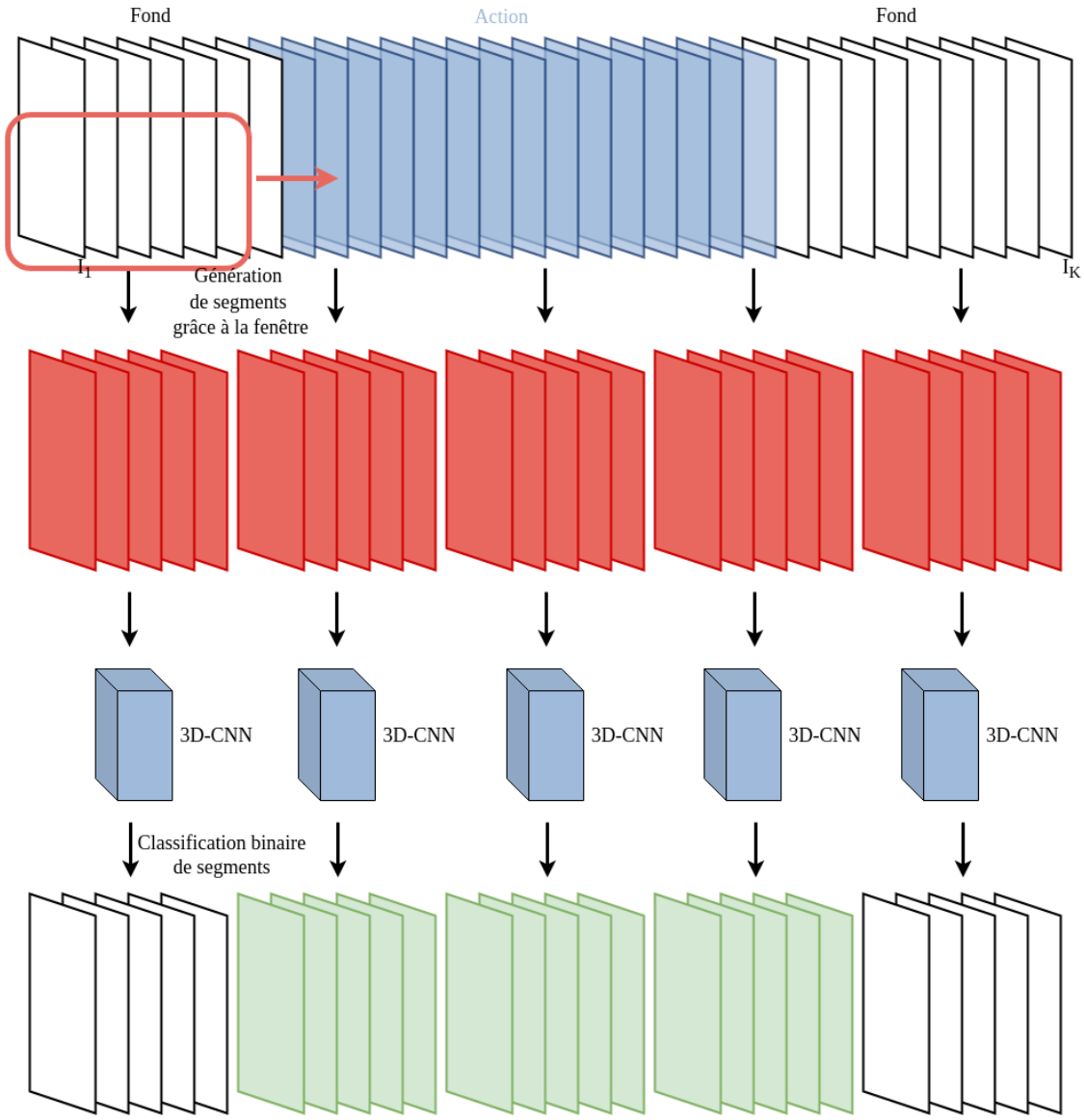


FIGURE 2.7 – **Schéma du principe de proposition temporelle d'actions par sliding window.**

Long *et al.* (2019) présente une architecture : GTAN (*Gaussian Temporal Awareness Networks*) qui redéfinit les ancres. Elle utilise des noyaux gaussiens pour prédire des propositions de régions d'action et estime ensuite la région finale. Des papiers tels que Chao *et al.* (2018); Xu *et al.* (2017) reprennent le faster R-CNN (*Region with Convolutional Neural Networks*) pour la détection d'objets présenté dans Ren *et al.* (2015), qu'ils adaptent à la détection de région d'action. Dans ces papiers, ils présentent le problème de localisation d'action comme la version 1D (temporelle) du problème de détection d'objets.

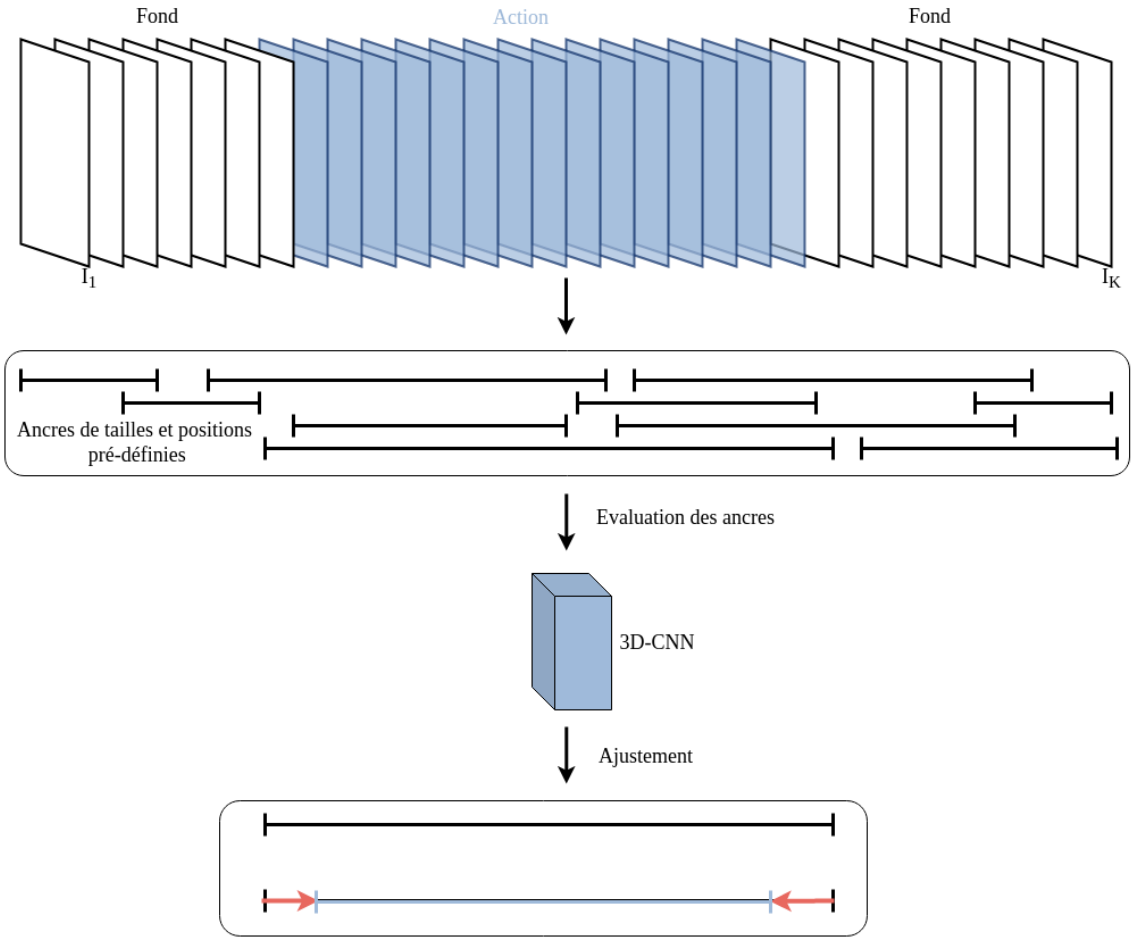


FIGURE 2.8 – **Schéma du principe d'ancres pour la proposition temporelle d'action.**

Score d'action

Les méthodes basées sur le calcul d'un score d'action (*Actionness Score*), qui peuvent également être appelées méthodes ascendantes (*bottom-up*) consistent à analyser les images une par une (ou des échantillons appelés *snippets*) et à attribuer un score d'action à chaque image. Les propositions d'actions sont ensuite estimées grâce à ce score. La Figure 2.9 résume ce principe.

Ce type de méthodes est généralement utilisé lorsque les bornes prédites nécessitent plus de précision ou que les images de fond et d'action sont plus facilement différentiables. Les méthodes descendantes sont efficaces mais connues pour manquer de précision dans la localisation de bornes. Notamment, des papiers récents tels que Lin *et al.* (2021) et Yang *et al.* (2020) présentent des méthodes qui s'affranchissent de la définition d'ancres. Selon eux, l'utilisation d'ancres ajoute du biais lors de l'apprentissage sur la position et la longueur des séquences d'action dans la vidéo. Ils expliquent que cela diminue la flexibilité des modèles à prédire des positions variées. Xiong *et al.* (2017) présente une méthode basée sur le calcul d'un score : TAG (*Temporal Actionness Grouping*) qui prédit

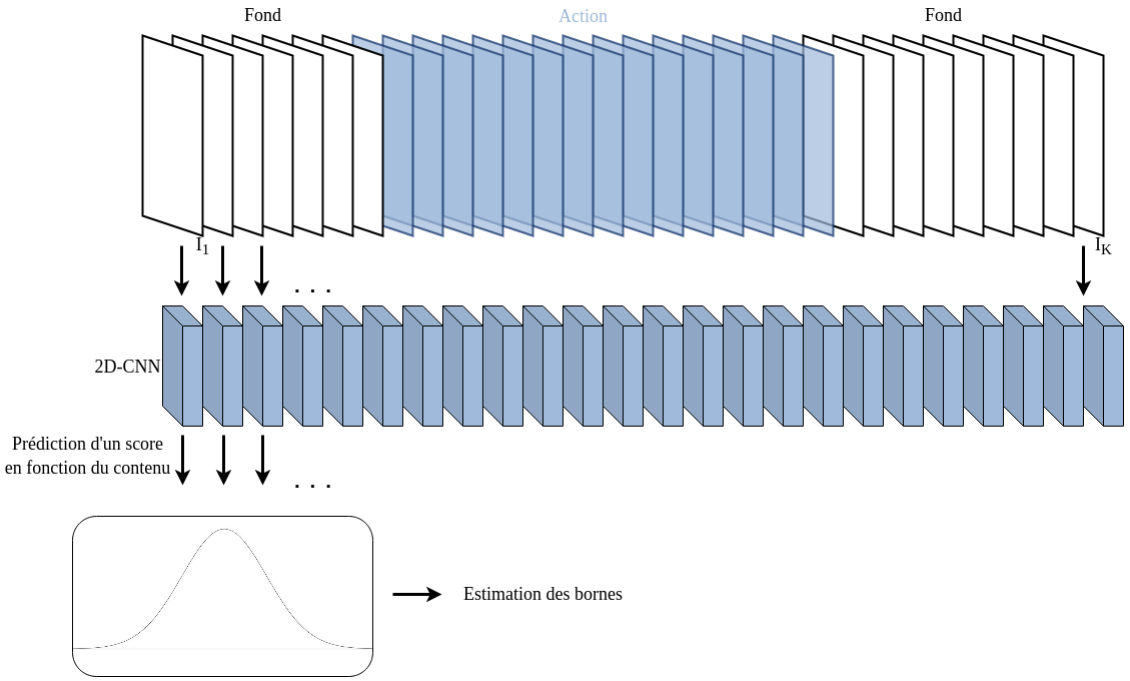


FIGURE 2.9 – **Schéma du principe de proposition temporelle d'actions par prédiction du score d'action.**

un score d'action sur des échantillons composés d'images et de flux optiques et propose des régions d'actions à partir de ce score sur la vidéo. Le score est établi pour chaque échantillon par un réseau pour la classification binaire appelé TSN (*Temporal Segment Network*). Cette méthode est construite ainsi pour améliorer la précision de la localisation en comparaison avec les méthodes basées sur les fenêtres glissantes. Ces méthodes sont pertinentes dans notre étude car elles sont très souvent appliquées au domaine du sport et sont d'autant plus efficaces sur des vidéos courtes où les images d'action et de fond sont facilement différenciables. Elles sont notamment utilisées pour localiser les moments où les sportifs performent et pour supprimer les autres moments, comme dans Lee *et al.* (2020).

2.2.2 Classification d'images

Pour pouvoir définir une méthode basée sur la prédiction d'un score d'action image par image, il est cohérent d'étudier le principe de classification d'images en fonction de leur contenu.

Débuts

En vision par ordinateur, le problème de classification d'images consiste à analyser une image pour lui associer une classe en fonction de son contenu. Il est présenté à la

fin des années 1950, avec l'arrivée des premières formes de réseaux de neurones, les perceptrons, définis par Rosenblatt (1958). Il utilise alors les perceptrons pour reconnaître et classifier des images de chiffres manuscrits. Des méthodes d'apprentissages automatiques et statistiques sont alors adaptées pour répondre à ce problème telles que les machines à vecteurs de support (SVM) présentés par Cortes et Vapnik (1995), les K plus proches voisins (KNN) adaptés à la classification par Cover et Hart (1967), les arbres de décisions, etc. En comparaison avec les réseaux de neurones, on appelle aujourd'hui ces approches "Méthodes traditionnelles". Depuis, l'évolution des réseaux de neurones et l'intérêt de cette thématique dans leur développement a fait exploser le nombre de méthodes pour la classification des images. Notamment, l'arrivée de concours de classification tels que ceux sur la base de données ImageNet, présentée par Deng *et al.* (2009) ont permis de participer à l'avènement des réseaux de neurones convolutifs et à la conception des méthodes les plus influentes de la vision par ordinateur comme le réseau AlexNet présenté par Deng *et al.* (2009).

Réseaux de neurones très profonds

C'est également la classification d'images qui a permis de montrer l'intérêt des réseaux très profonds. Plusieurs familles de réseaux de neurones pour la classification se sont alors distinguées. Les **VGG** (*Visual Geometry Group*) (Simonyan et Zisserman, 2014) sont l'une des premières architectures de réseaux convolutifs très profondes à se démarquer dans le domaine. L'utilisation d'architectures très profondes posait avant certains problèmes lors de l'apprentissage tels que le surapprentissage ou l'annulation du gradient. Dans ces travaux, ils parviennent à contrôler ces phénomènes en utilisant seulement des convolutions 3x3 qui limitent le nombre de paramètres même lorsque le nombre de couches devient très grand. He *et al.* (2016) présente ensuite les **ResNet**, qui sont capables d'aller encore plus profond. Pour ça, ils définissent l'architecture résiduelle. Cette architecture ajoute des connexions directes entre les couches. Cette opération permet de sauter certaines couches de convolution en connectant directement l'entrée d'un bloc résiduel à sa sortie, comme le montre la Figure 2.10. Dans ce papier, ils présentent notamment une architecture profonde de 34 couches (Figure 2.11), soit 15 de plus que les VGG-19. Ce principe a induit la conception de plusieurs architectures très performantes ensuite telles que **DenseNet** (Huang *et al.*, 2017) ou encore **ResNext** (Xie *et al.*, 2017).

Systèmes embarqués et transformers

Plus récemment, de nouvelles architectures plus rapides ont vu le jour pour pouvoir permettre la classification d'images dans les systèmes embarqués (Howard *et al.*, 2017;

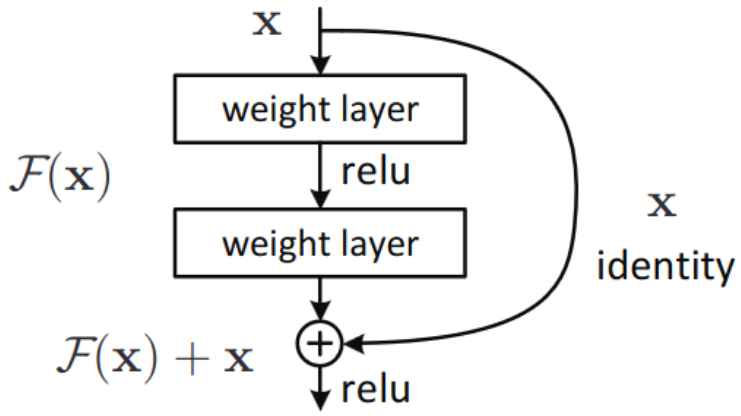


FIGURE 2.10 – **Schéma du principe d'apprentissage résiduel.** Ce schéma est tiré du papier He *et al.* (2016).

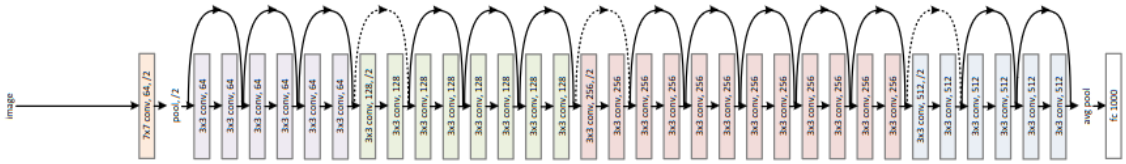


FIGURE 2.11 – **Architecture ResNet34.** Ce schéma est tiré du papier He *et al.* (2016).

Sandler *et al.*, 2018; Zhang *et al.*, 2018; Ma *et al.*, 2018). Les modèles basés sur les transformers pour le traitement du langage ont également été adaptés au traitement des images et à la classification dans Dosovitskiy *et al.* (2020). Ces modèles ont été une révolution dans ce domaine car ils permettent d'obtenir d'excellents résultats en accélérant considérablement les temps de calculs qui peuvent être longs avec l'utilisation de réseaux très profonds.

2.2.3 Position de nos travaux et contribution

Dans nos travaux, on étudie des vidéos courtes, où les régions d'action et de fond sont facilement différentiables. Le but étant la localisation précise des actions, ces éléments justifient de la définition d'une méthode ascendante qui classifie image par image le contenu de la vidéo et propose la région d'action à partir d'un score d'action.

La contribution de notre méthode se situe dans le principe de vérification/validation des bornes de localisation. Pour des méthodes de classification de données où les images sont très variables, elle permet d'obtenir un nombre de vidéos endommagées très bas tout en gardant un taux de régions bien localisées élevé. C'est un aspect peu abordé dans la littérature qui est pourtant très présent lors de la conception de produits finis. Ces travaux ont donné lieu à la publication présentée dans Baldanza *et al.* (2021).

2.3 Analyse du contenu par descripteurs

Dans cette partie, nous allons définir des descripteurs mathématiques qui permettent de discriminer les séquences d'images d'action des séquences de fond afin de permettre l'analyse de leurs variations au cours du temps. Ces descripteurs sont utilisés dans le processus de validation des propositions temporelles d'actions.

Les descripteurs présentés dans cette partie ont été définis et sélectionnés expérimentalement pour leurs performances sur la base de données présentée dans la Section 2.4.2. Un grand nombre de descripteurs (basés sur des principes tels que la variance des couleurs dans l'image, la répartition des hautes fréquences, des basses fréquences, les moyennes de patchs, ...) ont été testés mais n'ont pas donné de résultats suffisamment cohérents pour être retenus.

2.3.1 Descripteurs colorimétriques

On définit ici les descripteurs colorimétriques qui vont permettre de valider les bornes proposées.

Détection du ciel

Soit $N \in \mathbb{N}$ la largeur de l'image I_j. D_1 est défini comme la moyenne du canal couleur bleue de l'image dans la bande horizontale des $p \in \mathbb{N}$ pixels en haut de l'image :

$$D_1(j) = \frac{1}{p \times N} \sum_{x=1}^{N} \sum_{y=1}^{p} I_j^B(x, y). \tag{2.3}$$

On constate dans l'image 2.12 que la valeur de D_1 est nettement supérieure dans l'image de gauche que dans celle de droite. L'objectif de ce descripteur est de détecter le ciel dans l'image afin de différencier les images qui ne contiennent que le sol de celles où le ciel est présent.

Différence entre le haut et le bas de l'image

L'objectif de ce descripteur est d'analyser la différence d'intensité entre le haut de l'image et le bas, qui est plus petite lorsque l'image contient uniquement du sol ou du ciel que lorsque les deux sont répartis à l'intérieur. Soit $M \in \mathbb{N}$ la hauteur de l'image, D_2 calcule la différence de moyennes entre le premier et le dernier quart horizontal de

FIGURE 2.12 – **Calcul de la détection de ciel D_1 sur une image d'action et une image de fond.** Sur l'image de gauche, Sur cette image, $D_1(I) = 0.89$. Sur l'image de droite, $D_1(I) = 0.35$.

l'image I_j

$$D_2(j) = \frac{1}{N\frac{M}{4}} \sum_{x=1}^{N} \sum_{y=1}^{M/4} \sum_{C=\{R,G,B\}} I_j^C(x,y) - I_j^C\left(x, y + \frac{3M}{4}\right). \quad (2.4)$$

FIGURE 2.13 – **Calcul de la différence entre premier et dernier quart horizontal D_2 sur une image d'action et une image de fond.** Sur l'image de gauche, $D_2(I) = 0.72$. Sur l'image de droite, $D_2(I) = 0.09$. Le rectangle rouge représente le premier quart horizontal de l'image, et le rectangle bleu le dernier.

Dans la Figure 2.13, on constate aussi que D_2 est beaucoup plus élevé pour l'image de gauche que pour l'image de droite.

Différence d'histogrammes cumulés

Les transitions entre les sous-séquences d'action et de fond sont marquées par de fortes variations dans le contenu des images. Dans cette section, on définit donc un descripteur qui analyse la différence entre une image et la précédente. Pour ça, le descripteur calcule la différence d'histogrammes cumulés entre les images I_j et I_{j-1}. Soit

$H_c(I_j) \in \mathbb{R}^b$ l'histogramme cumulé calculé sur l'image I_j, b le nombre d'intervalles utilisés pour construire l'histogramme. On définit $d_{H_c}(j) \in \mathbb{R}$ la différence entre l'histogramme cumulé d'une image et celui de l'image précédente

$$d_{H_c}(j) = ||H_c(I_j) - H_c(I_{j-1})||_1. \tag{2.5}$$

Pour rendre ce descripteur plus robuste aux variations présentes à l'intérieur d'une même action, le descripteur final est défini de la façon suivante.

Soit G un noyau gaussien défini par

$$G(x, s) = \frac{1}{\sqrt{2\pi}s} e^{\frac{-x^2}{2s^2}}. \tag{2.6}$$

Soit

$$\begin{array}{rccc} \Phi & : & [\![1; K]\!] & \rightarrow & \mathbb{R} \\ & & j & \mapsto & \Phi(j) = (G * d_{H_c})(j) \end{array} \tag{2.7}$$

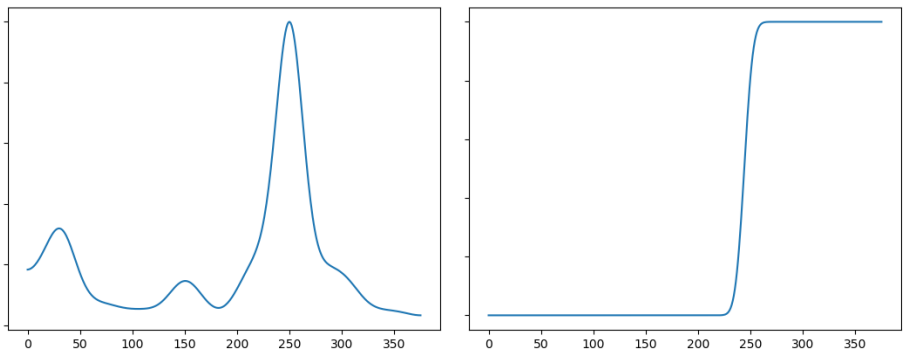
le résultat du produit de convolution entre d_{H_c} et le noyau gaussien G donné par (2.6).

On définit le descripteur D_3 de manière à normaliser ses variations en fonction de la valeur de $\Phi(j)$. On commence par définir D_3' par

$$D_3'(j) = \begin{cases} 1 & \text{si } |\Phi(j)| > \mu. \\ 0 & \text{sinon.} \end{cases} \tag{2.8}$$

avec $\mu \in \mathbb{R}$ un seuil fixé. Le descripteur D_3 est l'intégration de D_3' normalisée entre 0 et 1, c'est-à-dire

$$D_3(j) = \frac{\sum_{i=1}^{j} D_3'(i)}{\sum_{i=1}^{K} D_3'(i)}. \tag{2.9}$$



(a) (b)

FIGURE 2.14 – **Courbes obtenues à chaque étape du calcul.** (a) représente la courbe de d_{H_c} convolué avec un noyau gaussien. La courbe (b) est celle du descripteur final D_3.

Les courbes de la Figure 2.14 montrent que le descripteur D_3 est beaucoup plus précis et robuste aux légères variations qui ont lieu à l'intérieur des séquences d'actions et de fond.

2.3.2 Étude des variations

Le processus de validation de la proposition d'action est calculé en étudiant les variations des descripteurs au cours du temps. Ces descripteurs ont été définis pour discriminer les séquences d'actions des séquences de fond. Ainsi, on suppose la localisation temporelle qui maximise la variation d'un descripteur comme l'endroit de transition entre les deux sous-séquences. Ici, on va définir la fonction qui étudie ces variations pour trouver les bornes de validation.

Pour favoriser leur analyse temporelle, les descripteurs sont filtrés par produit convolutionnel avec un noyau gaussien. Soit $\delta_k(j_1, j_2) = \frac{1}{j_2 - j_1 + 1} \sum_{j=j_1}^{j_2} D_k(j)$ la fonction qui renvoie la moyenne du descripteur D_k sur une sous-séquence comprise entre $(j_1, j_2) \in [\![1; K]\!]^2$. On définit les bornes de variation $v_1^k, v_2^k \in [\![1; K]\!]^2$ issues du descripteur D_k comme les valeurs qui maximisent les différences de moyennes entre les sous-séquences du descripteur :

$$(v_1^k, v_2^k) = \underset{j_1, j_2}{\text{argmax}} \; ||\delta_k(1, j_1) - \delta_k(j_1, j_2)||^2 + ||\delta_k(j_2, K) - \delta_k(j_1, j_2)||^2. \tag{2.10}$$

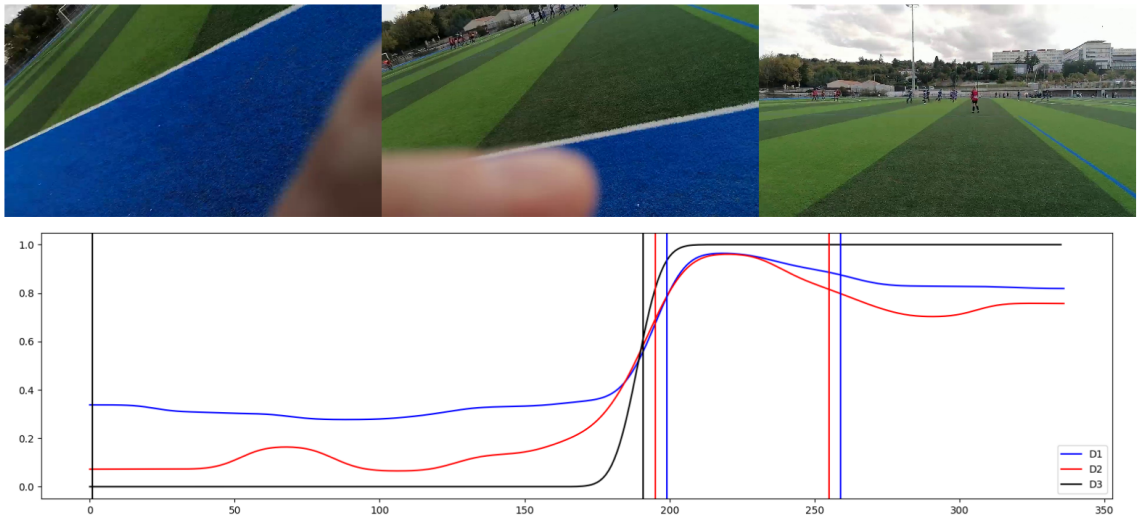


FIGURE 2.15 – **Variations des descripteurs sur une vidéo.** Les traits représentent les bornes de variations de chaque descripteur calculées en résolvant l'équation (2.10).

L'équation (2.10) est maximisée par recherche exhaustive. La Figure 2.15 présente un exemple de vidéo avec les courbes de chaque descripteur et les bornes de validation que leur étude fournit. Les bornes de variation calculées sont utilisées dans le processus de validation des bornes proposées par les méthodes que nous allons présenter dans la suite.

2.4 Proposition/validation temporelle d'action

Comme expliqué dans l'introduction, nos travaux ont donné naissance à deux méthodes. Dans ces travaux, les images correspondant aux séquences d'actions sont visuellement différentes des images de fond. Pour que nos méthodes soient les plus précises possibles, elles estiment la région d'action à partir d'un score calculé grâce à une classification image par image. Nous avons défini une première méthode qui calcule ce score grâce à la détection des joueurs dans l'image. La deuxième méthode classifie les images grâce à un réseau de neurones résiduel entraîné sur nos données. Nous détaillons les deux méthodes dans cette partie.

2.4.1 Première méthode : détecteur de joueurs

Dans cette étude, la présence de joueurs dans l'image est un critère de classification d'une séquence d'action. Détecter les personnes dans l'image nous donne une indication précise de la présence de joueurs dans celle-ci. Dans notre base de données, la plupart des sous-séquences de fond proviennent de mauvais cadrages de la personne qui filme. Elles sont donc principalement composées du sol ou du ciel, et ne contiennent que rarement des personnes.

Détection de personnes

La variabilité de la qualité des images dans nos environnements rend les méthodes de détection de personnes classiques peu robustes. Notamment, les méthodes qui utilisent les réseaux de neurones obtiennent des faux négatifs qui risquent d'endommager certaines vidéos à cause de leur résolution trop faible. Pour être plus robuste, la méthode combine la détection de personnes selon deux principes : les histogrammes de gradient orientés (HOG) (Dalal et Triggs, 2005) et les cascades de Haar (Viola et Jones, 2001b). La position, et le nombre de joueurs dans l'image n'est pas une information capitale. Nous cherchons simplement à détecter leur présence. Soit $l_{HOG}(I_j)$ et $l_{Haar}(I_j) \in \mathbb{N}$ respectivement le nombre de personnes détectées par ces méthodes, on définit

$$D_p(j) = \mathbb{1}_{l_{HOG}(I_j)+l_{Haar}(I_j)>0}, \tag{2.11}$$

avec $\mathbb{1}$ la fonction indicatrice définie par :

$$\mathbb{1}_{l>0} = \begin{cases} 1 & \text{si } l > 0. \\ 0 & \text{sinon.} \end{cases} \tag{2.12}$$

Le résultat de la détection est filtré par convolution avec un noyau gaussien pour atténuer l'impact des erreurs de détections et faciliter son analyse.

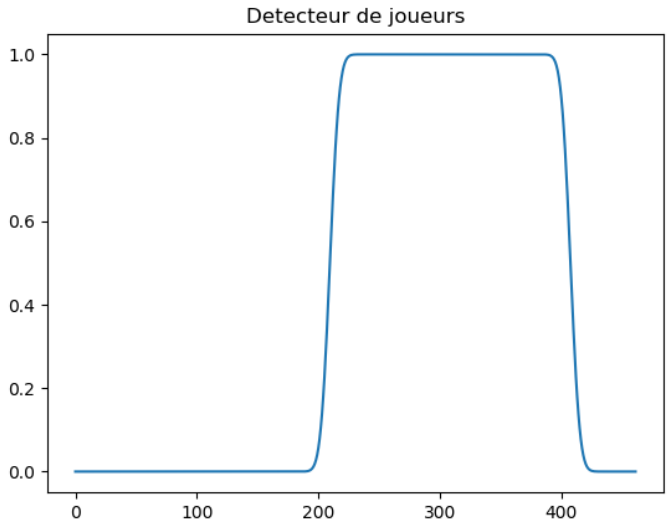


FIGURE 2.16 – **Score du détecteur de joueurs en fonction du temps.**

La Figure 2.16 montre les variations du détecteur de joueurs sur une vidéo composée de deux sous-séquences à supprimer : il ne détecte des joueurs qu'au milieu de la vidéo.

La Figure 2.17 montre des images dans lesquelles des joueurs sont détectés par les deux méthodes séparément. La ligne d'image du bas met en avant la complémentarité des deux méthodes : la mauvaise qualité de l'image rend la méthode de Haar moins performante, et seule la méthode basée sur les HoG a réussi à trouver des joueurs.

Proposition/validation des bornes

Notre méthode estime les bornes de la région d'action en étudiant les résultats du détecteur de joueurs. Comme pour les descripteurs colorimétriques, on cherche à localiser les deux endroits des plus fortes variations du détecteur au cours du temps. On définit $(p_1, p_2) \in [\![1; K]\!]^2$ les bornes de localisation temporelle calculée grâce à l'équation (2.10) que l'on calcule sur D_p via $\delta_p(j_1, j_2) = \frac{1}{j_2 - j_1 + 1} \sum_{j=j_1}^{j_2} D_p(j)$

HOG | Haar | Aucun joueur détecté

FIGURE 2.17 – **Résultats des détecteurs de personnes sur nos images.** À gauche, les joueurs sont détectés par la méthode HoG (Dalal et Triggs, 2005). Au milieu, les joueurs détectés par la méthode cascades de Haar (Viola et Jones, 2001b). À droite, des images dans lesquelles aucun joueur n'a été détecté par ces deux méthodes.

$$(p_1, p_2) = \underset{j_1, j_2}{\text{argmax}} \, ||\delta_p(1, j_1) - \delta_p(j_1, j_2)||^2 + ||\delta_p(j_2, K) - \delta_p(j_1, j_2)||^2. \tag{2.13}$$

On définit maintenant le processus de validation de la proposition.

On définit $(J_1, J_2) \in [\![1; K]\!]^2$ les bornes de localisation finales :

$$J_1 = \begin{cases} p_1 & \text{si } \min\limits_{\substack{i \in \{1,2\} \\ k \in \{1,\dots,3\}}} |p_1 - v_i^k| < \Delta. \\ 0 & \text{sinon.} \end{cases} \tag{2.14}$$

$$J_2 = \begin{cases} p_2 & \text{si } \min\limits_{\substack{i \in \{1,2\} \\ k \in \{1,\dots,3\}}} |p_2 - v_i^k| < \Delta. \\ K & \text{sinon.} \end{cases} \tag{2.15}$$

avec v_i^k les bornes de validation définies à l'équation (2.10) et Δ un seuil fixé expérimentalement à 35 pour cette méthode.

La Figure 2.18 schématise le pipeline de cette méthode.

2.4.2 Deuxième méthode : AxionNet

Dans cette méthode, la proposition temporelle est estimée sur une courbe d'action calculée par un réseau de neurones résiduel. Cette courbe est calculée grâce à une classification binaire image par image sur la vidéo. On applique ensuite le processus de vérification à cette proposition. On appelle cette méthode **AxionNet**. Pour pouvoir entraîner un

Algorithme 1 : Méthode 1 : Détection de joueurs

Entrées : V une vidéo de K images
pour $t = 1, ..., K$ **faire**
  Calculer $D_1(t), D_2(t), D_3(t), D_p(t)$;
fin
$p_1 = 0$, $p_2 = K$;
$v_1^1 = 0$, $v_2^1 = K$, $v_1^2 = 0$, $v_2^2 = K$, $v_1^3 = 0$, $v_2^3 = K$;
$c = 0$, $c^1 = 0$, $c^2 = 0$, $c^3 = 0$;
pour $j_1 = 1, ..., K$ **faire**
  pour $j_2 = j_1, ..., K$ **faire**
    Calculer $\delta_p(1, j_1), \delta_p(j_1, j_2), \delta_p(j_2, K)$;
    si $||\delta_p(1, j_1) - \delta_p(j_1, j_2)||^2 + ||\delta_p(j_2, K) - \delta_p(j_1, j_2)||^2 > c$ **alors**
      $p_1 = j_1$, $p_2 = j_2$;
      $c = ||\delta_p(1, j_1) - \delta_p(j_1, j_2)||^2 + ||\delta_p(j_2, K) - \delta_p(j_1, j_2)||^2$;
    sinon
    fin
    pour $k = 1, 2, 3$ **faire**
      Calculer $\delta_k(1, j_1), \delta_k(j_1, j_2), \delta_k(j_2, K)$;
      si $||\delta_k(1, j_1) - \delta_k(j_1, j_2)||^2 + ||\delta_k(j_2, K) - \delta_k(j_1, j_2)||^2 > c^k$ **alors**
        $v_1^k = j_1$, $v_2^k = j_2$;
        $c^k = ||\delta_k(1, j_1) - \delta_k(j_1, j_2)||^2 + ||\delta_k(j_2, K) - \delta_k(j_1, j_2)||^2$;
      sinon
      fin
    fin
  fin
fin
$J_1 = 0$, $J_2 = K$;
pour $k = 1, 2, 3$ **faire**
  pour $i = 1, 2$ **faire**
    si $|p_1 - v_i^k| < \Delta$ **alors**
      $J_1 = p_1$;
    sinon
    fin
    si $|p_2 - v_i^k| < \Delta$ **alors**
      $J_2 = p_2$;
    sinon
    fin
  fin
fin
Sorties : J_1, J_2

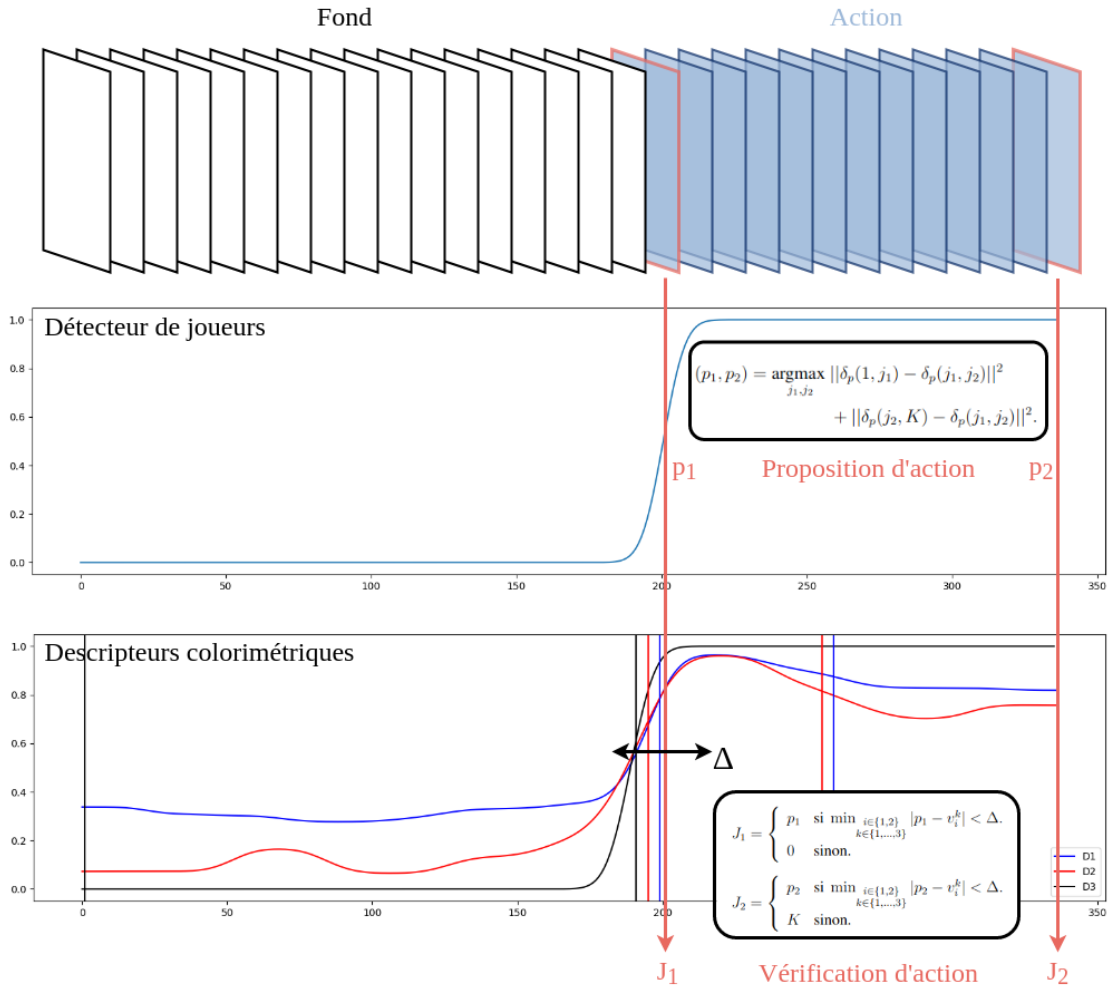


FIGURE 2.18 – **Schéma explicatif de la méthode basée sur la détection de joueurs.**

tel réseau, et qu'il soit robuste aux variations présentées par les vidéos de sport amateur, la base de données doit être diversifiée et suffisamment grande.

Base de données et annotation

Pour entraîner un réseau de neurones robuste aux variations dans les vidéos de sport amateur (qualité de la caméra, point de vue de captation, terrains, environnements, météo, ...), la base de données doit être diversifiée et grande. L'annotation manuelle d'une telle base de données est une tâche longue et fastidieuse. Pour accélérer ce processus, nous avons utilisé la méthode définie dans la Section 2.4.1. En calculant la méthode précédente sur toutes les vidéos de la base de données d'apprentissage, on procède à une pré-labélisation des données. Les images prélevées dans les séquences à l'extérieur des bornes d'action prédites sont pré-annotées comme "Fond", et les autres comme "Action". Il suffit ainsi de contrôler que les images ont bien été traitées par l'algorithme et de modi-

fier les mauvaises annotations pour construire la base d'apprentissage annotée finale.

De cette façon, nous avons construit une base de données composée de 61043 images provenant de 1940 vidéos différentes pour l'apprentissage du réseau. Ces vidéos sont issues de 4 sports différents (basket-ball, handball, football et rugby) pour que la classification soit robuste aux variations induites par les sports. En comparant expérimentalement les performances de classification, nous avons remarqué que les réseaux entraînés sur les images des sports en salle et en extérieur séparément obtenaient de meilleurs résultats que lorsqu'ils étaient entraînés sur toute la base de données. La grande différence de contexte entre les deux environnements explique ce type de résultat.

La base de données a donc été divisée en deux (30800 images en salle et 30243 images en extérieur), pour pouvoir entraîner deux réseaux séparés.

Architecture

Pour la classification des images, on utilise une architecture résiduelle ResNet-18 présentée dans He *et al.* (2016) pré-entraînée sur la base de données ImageNet (Russakovsky *et al.*, 2015). L'architecture de ce réseau est présentée dans la Figure 2.19

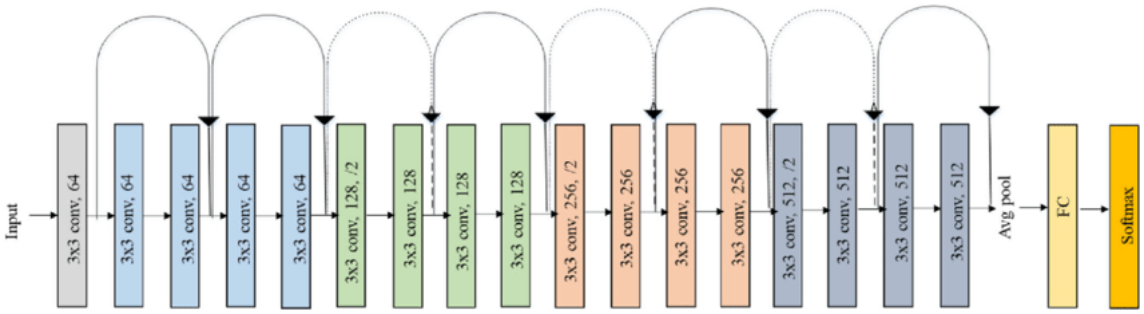


FIGURE 2.19 – **Schéma de l'architecture originale ResNet-18.** Ce schéma est tiré du papier Ramzan *et al.* (2019).

La dernière couche du réseau pré-entraîné destinée à classifier les images parmi les 1000 classes de la base de données d'ImageNet est remplacée par une couche pour la classification binaire de notre étude. L'utilisation d'une telle architecture permet à notre modèle de bénéficier de l'extraction de caractéristique et des représentations de haut niveau des images apprises sur une base de données vaste et très variées. Ceci est impossible sur notre base de données d'apprentissage. Elle permet également d'entraîner la dernière couche qui permet la classification des images sur nos données. Cette architecture a été sélectionnée expérimentalement car elle obtient de très bons résultats pour la classification et reste rapide à calculer. Elle permet au pipeline complet d'être calculé en moins de 10s sur toutes les vidéos de la base.

Proposition/validation des bornes

Ici aussi, la proposition d'action est estimée en résolvant l'équation (2.10) sur la courbe d'action.
On définit ici $D_c(j)$ le résultat de la classification au moment j. On calcule les bornes $(p_1, p_2) \in [\![1; K]\!]^2$ de proposition temporelle via $\delta_c(j_1, j_2) = \frac{1}{j_2 - j_1 + 1} \sum_{j=j_1}^{j_2} D_c(j)$ comme précédemment

$$(p_1, p_2) = \underset{j_1, j_2}{\text{argmax}} \; ||\delta_c(1, j_1) - \delta_c(j_1, j_2)||^2 + ||\delta_c(j_2, K) - \delta_c(j_1, j_2)||^2. \tag{2.16}$$

On applique à cette proposition le processus de validation définie dans (2.14) et (2.15) que l'on réécrit :

$$J_1 = \begin{cases} p_1 & \text{si } \min\limits_{\substack{i \in \{1,2\} \\ k \in \{1,\dots,3\}}} |p_1 - v_i^k| < \Delta. \\ 0 & \text{sinon.} \end{cases} \tag{2.17}$$

$$J_2 = \begin{cases} p_2 & \text{si } \min\limits_{\substack{i \in \{1,2\} \\ k \in \{1,\dots,3\}}} |p_2 - v_i^k| < \Delta. \\ K & \text{sinon.} \end{cases} \tag{2.18}$$

Le pipeline complet de cette méthode est détaillé dans la Figure 2.20. Le seuil Δ a été expérimentalement fixé à 60 pour permettre de diminuer significativement le nombre de faux négatifs en maximisant les performances de localisation. Choisir un seuil aussi large permet de rester robuste à un maximum de vidéos (la transition entre deux séquences peut prendre plus d'une seconde) et permet de limiter l'impact négatif de la validation sur la précision du modèle.

2.5 Résultats

Nos méthodes ont été conçues pour résoudre le problème de proposition temporelle d'actions sur notre base de données. Il est difficile d'entraîner les modèles de la littérature généralement évalués sur les bases de données présentées dans l'état de l'art, notamment car notre annotation est différente des leurs. De plus, certains choix architecturaux (tels que la classification image par image) ont été faits pour répondre aux objectifs des vidéos courtes de sport amateur. Il n'est donc pas cohérent de tester nos modèles sur les bases de données de la littérature.

Dans cette partie, on présentera nos résultats et on les comparera avec la méthode

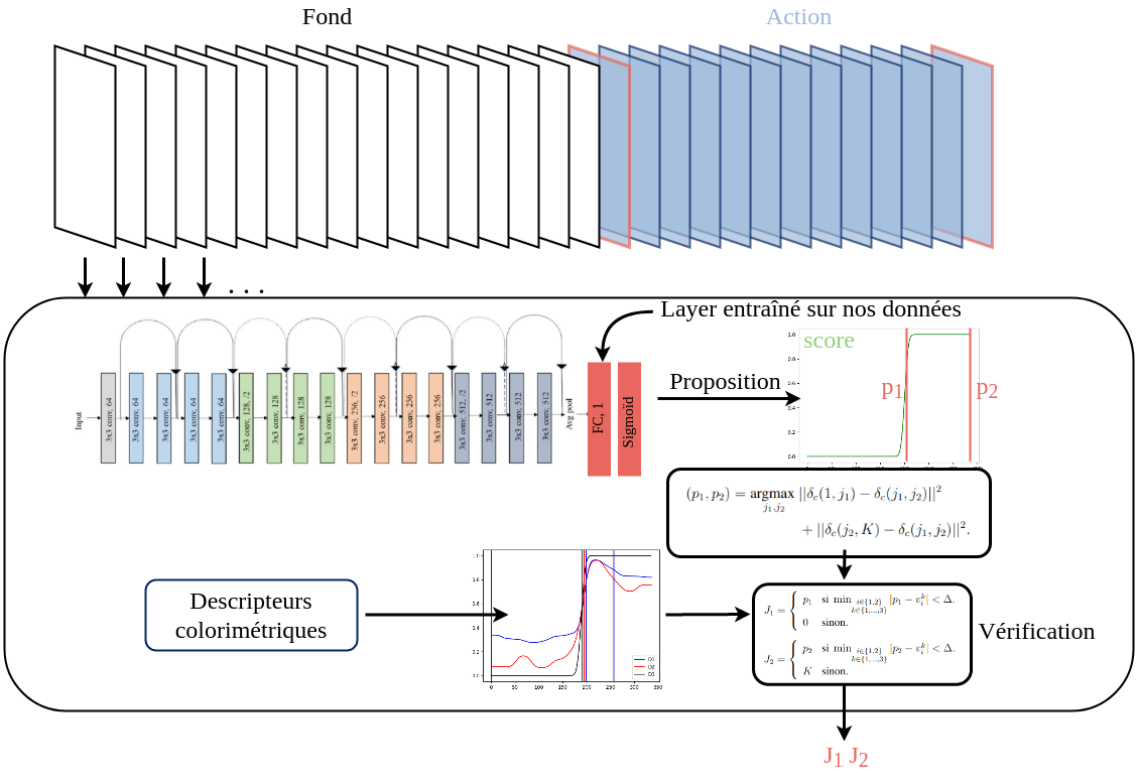


FIGURE 2.20 – **Schéma du pipeline de la deuxième méthode utilisant ResNet-18.** Cette image schématise la deuxième méthode de proposition/vérification temporelle d'actions.

sans le processus de validation pour justifier de notre contribution. L'intérêt est d'étudier la précision de nos modèles et de mettre en avant l'impact de cette validation sur une méthode de proposition temporelle d'actions. Les tests sont réalisés sur une base de données de 9328 vidéos issues de la plateforme Rematch. Cette base est construite de la même façon que la base de données d'apprentissage présentée dans la Section 2.4.2. On évalue les modèles sur les 4 sports utilisés pour l'entraînement. Dans cette base de données, 4517 vidéos sont filmées en extérieur, et 4811 en salle. L'évaluation de nos travaux est réalisée en deux temps. Dans un premier temps, nous évaluons les performances des modèles à localiser l'action dans des vidéos non-bornées (Définition 1.2.2). La base de données contient 467 vidéos de ce type. Cette étude permet d'évaluer la précision des modèles pour la localisation, ainsi que l'impact négatif que peut avoir le processus de validation sur celle-ci. Ensuite, pour mettre en avant l'impact positif du processus de validation, et étudier la capacité des modèles à ne pas endommager les vidéos, on étudie les performances sur toute la base de données (vidéos bornées et non-bornées).

2.5.1 Métrique d'évaluation

Les travaux de détection d'actions tels que Jiang *et al.* (2014); Fabian Caba Heilbron et Niebles (2015) utilisent le **mAP** (*Mean Average Precision*) comme métrique d'évaluation pour leurs tâches. Cette métrique calcule la moyenne pour chaque classe de la précision moyenne du modèle. Or, dans nos travaux, nous ne cherchons pas à classifier l'action. Calculer le mAP revient donc à calculer la précision moyenne (AP). Comme expliqué dans l'introduction de ce chapitre, ces métriques sont calculées pour un seuil de précision appelé tIoU (*temporal Intersection over Union*), et dont la formule est donnée à l'équation (2.1). Pour qu'une action prédite soit considérée comme une bonne localisation, appelée "vrai positif", il faut qu'elle soit suffisamment proche, au sens du seuil tIoU, de la vérité terrain. Dans notre étude, nous cherchons à évaluer les modèles pour un seuil tIoU proche de 1, afin de s'assurer que le modèle n'endommage pas les séquences d'action. On évaluera donc le mAP pour un tIoU à 0.8 (mAP@0.8).

Mean Average Precision (mAP)

Pour calculer le mAP, on utilise deux mesures : la *Précision*, qui mesure la proportion de vrais positifs qui sont effectivement corrects, et le *Rappel*, qui mesure la proportion de positifs réels qui ont été prédits correctement. Ces mesures sont calculées de cette manière :

$$Precision = \frac{TP}{TP + FP} \tag{2.19}$$

$$Rappel = \frac{TP}{TP + FN} \tag{2.20}$$

où TP représente le nombre de vrais positifs, FP les faux positifs et FN les faux négatifs. Ces deux mesures sont comprises entre 0 et 1. En calculant ces deux mesures après chaque vidéo, on peut tracer la courbe de précision-rappel qui représente la précision $p(r)$ en fonction du rappel r. Le mAP est calculé comme l'aire sous cette courbe.

$$mAP = \int_0^1 p(r)\mathrm{d}r \tag{2.21}$$

Pour calculer cette aire, on utilisera la méthode d'interpolation sur 11 points présentée dans Everingham *et al.* (2009) qui définit la formule suivante :

$$mAP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \ldots, 1\}} p_{interp}(r) \tag{2.22}$$

où $p_{interp}(r) = \max_{\tilde{r}:\tilde{r}>r} p(\tilde{r})$

On utilisera donc cette métrique pour évaluer la précision des modèles sur les vidéos non-bornées.

Pour évaluer les modèles en tant que méthodes de découpage automatique, on étudiera également les performances des méthodes sur toute la base de données, en évaluant le nombre de vidéos correctement traitées (i.e. vidéos non-bornées correctement localisées, et vidéos bornées non-endommagées).

2.5.2 Évaluation

Vidéos non-bornées

Les résultats obtenus sur la base de vidéos non-bornées sont présentés dans le Tableau 2.1. Les résultats obtenus par le modèle utilisant la détection de joueurs (première méthode à avoir été conçue) ont montré des performances encourageantes pour répondre au problème. L'utilisation du processus de validation permet au modèle de limiter fortement les faux négatifs obtenus par le détecteur de joueurs en n'altérant que légèrement la précision du modèle sur les vidéos non-bornées. L'observation de ces résultats a mené à l'intuition qu'une méthode basée sur un réseau de neurones entraîné spécifiquement pour cette classification pouvait donner d'excellentes performances.

Cette intuition est vérifiée lorsqu'on regarde les résultats obtenus par la méthode AxionNet (avec et sans le processus de validation). La méthode basée sur l'utilisation de l'architecture ResNet-18 qui n'utilise pas le processus de validation obtient d'excellents résultats de localisation (mAP = 97.7% en extérieur et 92.6% en salle). La méthode AxionNet utilise le même processus de proposition. Le processus de validation ne permettant que d'annuler des découpes, il parrait logique qu'elle n'obtienne pas de meilleurs résultats. Il est cependant intéressant de constater que le processus de validation n'altère que très peu sa précision puisque le modèle AxionNet obtient des résultats très proches sur cette base de données. Il atteint 95.6% de mAP en extérieur et 89.5% en salle. Par contre, les résultats de cette approche sont nettement supérieurs à ceux présentés par la première méthode qui utilise le détecteur de joueurs. Ce résultat s'explique car le réseau est spécifiquement entraîné pour cette tâche de classification. De plus, il peut arriver que des personnes soient présentes dans des séquences de fond, rendant le détecteur de joueurs incapable de différencier les séquences.

Impact sur le nombre de vidéos endommagées

Les Tableaux 2.2 et 2.3 présentent les résultats sur toute la base de données, contenant des vidéos bornées et non-bornées. Observer les résultats des modèles sur cette base permet de mettre en avant l'intérêt du processus de validation. Dans le Tableau 2.2,

mAP(%)@tIoU=0.8	Extérieur	Salle
Méthode 1 (Détecteur de joueurs)	77.7	78.8
AxionNet (Sans validation)	**97.7**	**92.6**
AxionNet	95.6	89.5

TABLEAU 2.1 – **Résultats des modèles sur les vidéos non-bornées.** Mesurés en mAP(%) pour un tIoU = 0.8. Le modèle **AxionNet** avec le processus de validation obtient des résultats proches de la version sans validation. Cela signifie que ce processus n'altère que peu la précision de la localisation.

on constate que la version d'AxionNet qui utilise le processus de validation obtient un nombre de vidéos endommagées inférieur. Notamment, AxionNet endommage 17 vidéos de moins que la version de l'algorithme sans le processus de validation. Dans le domaine du sport, 17 vidéos endommagées peuvent représenter un réel problème si une de ces vidéos contient une action d'intérêt majeur. Une telle amélioration est donc considérable. De plus, lorsqu'on regarde les vidéos endommagées, on s'aperçoit que le processus de validation permet également de réduire la taille des séquences endommagées. En effet, la méthode sans validation obtient une durée moyenne de séquence endommagée de 4,09s. Pour AxionNet cette moyenne est de 3,51s.

Vidéos endommagées	Extérieur	Salle	Total
Méthode 1 (Détecteur de joueurs)	52	29	81
AxionNet (Sans validation)	38	21	59
AxionNet	**27**	**15**	**42**

TABLEAU 2.2 – **Nombre de vidéos endommagées sur toute la base de données en fonction de la méthode de localisation utilisée.** Le modèle **AxionNet** est le modèle qui endommage le moins de vidéos.

Compromis entre précision et nombre de vidéos endommagées

L'intérêt de la méthode est donc qu'elle permet de minimiser le nombre de vidéos endommagées (Tableau 2.2) tout en conservant une précision de localisation performante (Tableau 2.1). En évaluant les performances sur la base de données entière, on constate que AxionNet présente un pourcentage de vidéos bien traitées légèrement supérieur à la version sans validation et largement au dessus de la méthode qui utilise le détecteur de joueurs (Tableau 2.3). Les vidéos bornées (qui contiennent uniquement de l'action) sont plus faciles à traiter. L'absence de ce type de vidéo dans la base de données de test du Tableau 2.1 justifie l'écart de performances entre les résultats du Tableau 2.1 et ceux du tableau 2.3. De plus, la méthode AxionNet permet de limiter fortement le nombre de vidéos endommagées. Notre méthode permet donc de réaliser un compromis bénéfique

entre la précision du modèle sur les vidéos non-bornées et sa capacité à ne pas détruire de séquences d'action. Un intérêt capital du processus de validation est qu'il peut être adapté à d'autres méthodes de proposition temporelle d'actions pour limiter l'impact des faux négatifs. Il suffit pour ça qu'il soit modifié pour correspondre à la discrimination souhaitée.

Vidéos bien coupées (%)	Extérieur	Salle	Total
Méthode 1 (Détecteur de joueurs)	97,87	98,06	97.97
AxionNet (Sans validation)	99.02	**99.02**	99.01
AxionNet	**99.20**	98.98	**99.08**

TABLEAU 2.3 – **Pourcentage de vidéos bien découpées en fonction de la méthode de localisation utilisée sur toute la base de données.** Le modèle **AxionNet** obtient le meilleur pourcentage de réussite.

Limites

Lorsqu'on observe les résultats dans leur ensemble, on s'aperçoit que le processus de validation a un impact légèrement plus performant en extérieur qu'en salle. Dans le Tableau 2.1, le mAP diminue de 2.1% en extérieur avec le processus de validation, et de 3.1% en salle. Dans le Tableau 2.2, le nombre de vidéos endommagées baisse de 11 vidéos en extérieur et de seulement 6 vidéos en salle. Ce résultat peut s'expliquer par la façon dont nous avons défini les descripteurs, qui semblent être plus adaptés aux sports en extérieur (notamment la détection du ciel).

2.6 Conclusion du chapitre

Dans ce chapitre, nous avons présenté des méthodes de proposition temporelle d'actions pour le découpage automatique des vidéos de sport amateur. Ces méthodes visent à supprimer les morceaux des vidéos qui ne contiennent pas d'action. Pour répondre à ce problème, nous avons conçu un système en deux temps. Dans un premier temps, notre méthode estime une proposition d'action sur un score prédit en fonction du contenu sur chaque image. Dans un second temps, la méthode utilise un processus de validation des bornes proposées qui a pour but de limiter le nombre de vidéos endommagées par la découpe tout en préservant la précision du modèle. Ce système est utilisé dans deux méthodes que nous avons évaluées pour répondre au problème. La première méthode estime la proposition à partir d'un score calculé grâce à la détection des joueurs dans l'image. La deuxième prédit ce score grâce à un réseau de neurones entraîné pour la classification binaire des images d'action. Les résultats ont montré que notre méthode permettait

d'obtenir des localisations d'actions précises sur les vidéos non-bornées tout en limitant fortement le nombre de faux-négatifs qui peuvent endommager les vidéos. L'intérêt de ce processus est qu'il peut aussi être adapté pour d'autres méthodes de proposition temporelle d'actions qui cherchent à limiter ce phénomène. Une des limites de notre méthode se situe dans la définition des descripteurs colorimétriques qui manquent parfois de robustesse dans des environnements très variables. Il serait intéressant de développer de nouvelles approches pour la validation (réseau entraîné pour cette tâche, ...) qui seraient plus robustes. La méthode **AxionNet** est aujourd'hui utilisée dans le processus d'encodage vidéo de la plateforme Rematch pour la découpe automatique des vidéos. Chaque vidéo capturée sur l'application est traitée par l'algorithme et les séquences situées à l'extérieur des bornes d'actions sont supprimées pour que les vidéos disponibles ne contiennent que l'action.

Chapitre 3

Suivi d'action pour caméra autonome : modèle linéaire par morceaux

Sommaire

3.1 Introduction 52
3.1.1 Motivations 53
3.1.2 Notre méthode 55
3.2 État de l'art 55
3.2.1 Suivi d'objets 56
3.2.2 Suivi d'objets dans les vidéos de sport 60
3.2.3 Positionnement de notre méthode 62
3.3 Flux optique et segmentation des plans 63
3.3.1 Segmentation premier plan/arrière-plan 64
3.3.2 Normalisation 65
3.4 Modèle de prédiction linéaire par morceaux 66
3.4.1 Modèle linéaire 66
3.4.2 Modèle linéaire par morceaux 67
3.4.3 Apprentissage et situations 68
3.5 Bases de données et annotations 69
3.5.1 Bases de données 69
3.5.2 Annotation 70
3.6 Résultats 70
3.6.1 Métrique d'évaluation 71
3.6.2 Évaluation 72
3.7 Conclusion du chapitre 76

3.1 Introduction

L'objectif de cette partie est de concevoir une méthode de suivi d'action vidéo pour permettre la capture autonome du sport amateur à partir d'une **caméra PTZ** (*Pan-Tilt-Zoom*). Les spécificités de ce type de caméra sont définies dans la Définition 1.2.3. La méthode présentée dans ce chapitre a pour but de suivre l'action du match à partir de l'analyse des images capturées par le flux unique de la caméra placée au bord du terrain. La Figure 3.1 schématise notre problématique. À chaque instant t, l'objectif est de prédire le mouvement d^t de la caméra nécessaire à suivre l'action.

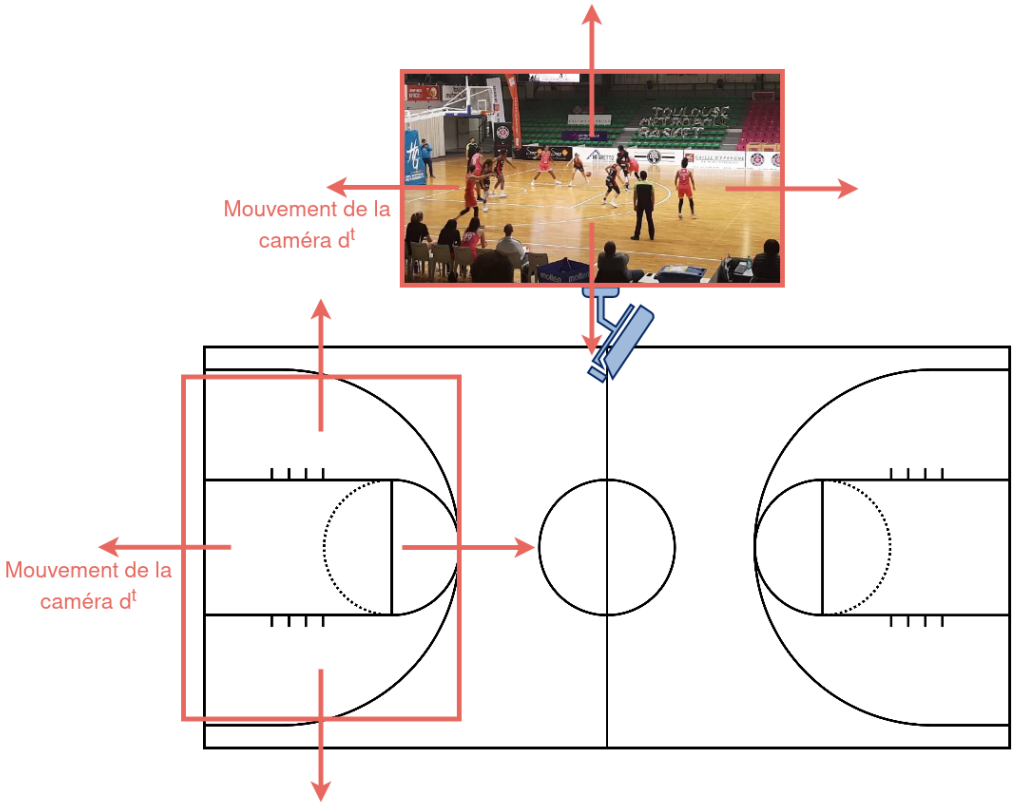


FIGURE 3.1 – **Schéma de l'objectif de notre méthode.** À partir de l'analyse du flux unique de la caméra PTZ placée au bord du terrain, on cherche à prédire à chaque instant t, les déplacements de la caméra d^t nécessaires à suivre l'action de jeu.

L'évaluation des méthodes de contrôle de caméras autonomes est une tâche complexe car chaque image capturée dépend du déplacement précédemment prédit. Certaines méthodes de la littérature qui évaluent cette tâche utilisent des images issues de caméras "virtuelles" générées en fonction des mouvements prédits. Par exemple, Papadakis *et al.* (2010) présente une méthode de synthèse d'images virtuelles pour des matchs de football qui peut être utilisée dans ce type d'applications pour générer les images en fonction des prédictions précédentes. D'autres méthodes telles que celles présentées dans Haque *et al.*

(2013) et Kumar *et al.* (2009) ont la possibilité de tester leurs modèles en flux continu directement avec des caméras PTZ. Cependant, dans la littérature, il n'existe pas ou peu de données disponibles issues de vidéos de sport amateur. Dans ce chapitre, nous ne disposons donc que de vidéos réelles déjà capturées disponibles sur la plateforme Rematch. Nous testerons le modèle par une évaluation hors ligne. Notre méthode prend en entrée une vidéo de sport amateur et prédit à chaque instant t de la vidéo les mouvements de la caméra nécessaires à suivre l'action du match. La prédiction à un instant t est réalisée en temps réel à partir des images précédemment acquises. La Figure 3.2 schématise la façon dont on applique notre méthode sur nos vidéos. L'intérêt est d'évaluer notre modèle en imitant les conditions des algorithmes de contrôle autonomes des systèmes embarqués, dans lesquels les prédictions ne peuvent pas dépendre d'images capturées dans le futur.

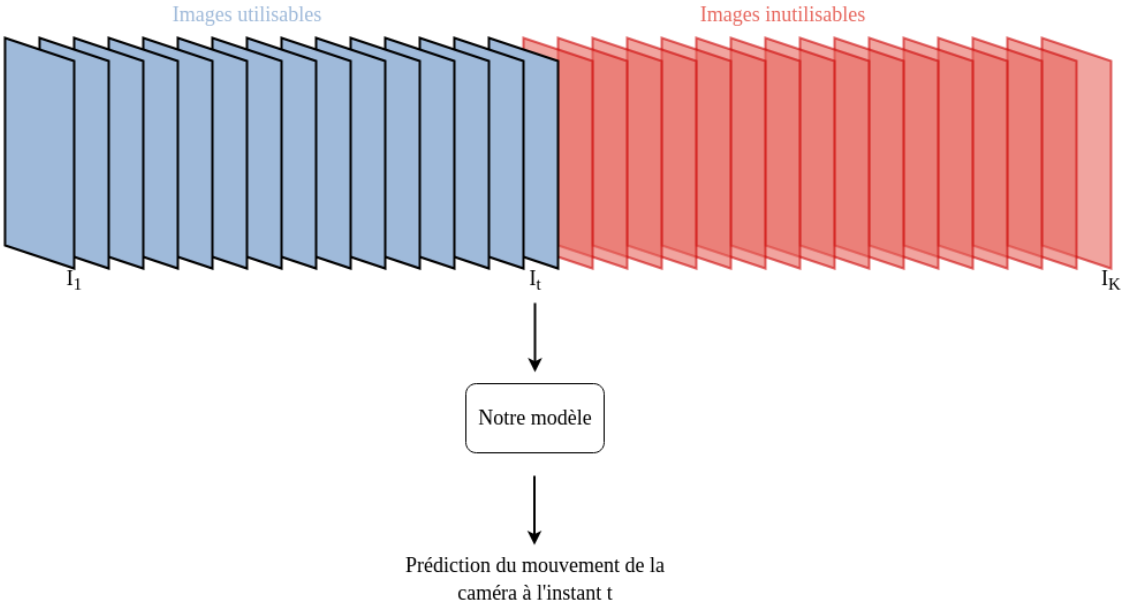


FIGURE 3.2 – **Schéma d'application de notre méthode sur les vidéos de nos bases de données.** Lors de la prédiction des mouvements de la caméra à un moment t de la vidéo, les images utilisables pour réaliser la prédiction sont celles déjà capturées à ce moment-là.

3.1.1 Motivations

Dans le sport professionnel, les caméras sont placées en hauteur et au même endroit dans tous les stades ou gymnases. Un tel placement permet de limiter l'occlusion de la balle et des joueurs ainsi que de diminuer les variations d'environnement d'un match à l'autre. Ces caractéristiques, ajoutées à la qualité des images capturées par les caméras et à la puissance des processeurs utilisés dans le milieu professionnel, permettent d'utiliser des méthodes de suivi par détection (*tracking-by-detection*). Le problème de suivi d'action revient donc à suivre le ballon et/ou les joueurs et à diriger la caméra dessus. Dans le

sport amateur, les données disponibles sont plus rares et généralement capturées par des bénévoles avec des caméras hétérogènes. La position de la caméra et la disposition des stades et gymnases varient d'un match à l'autre. Les angles de prise de vue sont généralement plus proches du sol et donc plus sensibles aux problèmes d'occlusion. Lorsqu'on ajoute à ces caractéristiques les qualités hétérogènes des caméras d'acquisition et la faible puissance de calcul des processeurs utilisables, il devient compliqué de définir et d'entraîner des méthodes de suivi par détection robustes pour diriger la caméra. Notamment, détecter le ballon devient une tâche impossible. Nous avons testé sur nos vidéos deux architectures pré-entraînées pour la détection d'objets en temps réel régulièrement utilisées dans des pipelines de suivi : **YOLO** (*You Only Look Once*) (Redmon *et al.*, 2016; Jocher *et al.*, 2022) et **Faster R-CNN** (*Regions with CNN features*) (Massa et Girshick, 2018). Ces tests avaient pour but d'évaluer leur capacité à détecter la balle dans nos images. Les résultats de ces tests sont présentés dans la Section 3.2.3. Elles sont toutes les deux parvenues à détecter le ballon dans moins de 15% des images. Un tel résultat est inutilisable pour concevoir une méthode de suivi par détection. Les Figures 3.3, 3.4, 3.5 et 3.6 illustrent ces résultats et mettent en avant les différentes difficultés que peut rencontrer la détection du ballon sur nos images.

FIGURE 3.3 – **Exemples d'images dans lesquelles on parvient à détecter le ballon.**

FIGURE 3.4 – **Exemples de fausses détections dans nos images.**

Burić *et al.* (2018) présente également les performances de différentes architectures pour la détection et le suivi d'objets sur des vidéos de sport amateur. Ce papier met lui aussi en avant les difficultés de ces méthodes à obtenir des résultats robustes. Dans la suite de cette thèse, nous allons chercher à concevoir une méthode nouvelle qui permet de traiter les problématiques liées au suivi d'action amenée par le contexte amateur.

FIGURE 3.5 – **Exemples d'images où le ballon n'est pas détecté.**

FIGURE 3.6 – **Exemples d'images où le ballon est caché et donc pas détectable.**

3.1.2 Notre méthode

Pour répondre aux problématiques amenées par le contexte du sport amateur, nous redéfinissons dans nos travaux la notion de suivi d'action. Dans la suite de la thèse, on définira l'action comme l'endroit du terrain vers lequel un humain dirige la caméra lors d'un match. L'idée derrière cette définition est de concevoir un modèle qui apprend et reproduit le comportement humain d'acquisition du sport. Pour cela, notre méthode vise à analyser les mouvements dans l'image au lieu de baser la prédiction de l'endroit à filmer sur le suivi d'objets. Nos travaux suivent l'intuition que les mouvements des joueurs dans l'image donnent une indication suffisante pour pouvoir suivre l'action du match. Comme expliqué dans la Section 1.2, le contrôle de caméra autonome est généralement séparé en deux modules : le module de suivi et le module de contrôle. Dans nos travaux, nous allons proposer un pipeline à un étage qui déduit de l'analyse de l'image le déplacement de la caméra nécessaire pour suivre l'action, sans passer par une localisation spatiale dans l'image. Ce chapitre présente une méthode linéaire par morceaux pour calculer cette prédiction. Ces travaux ont donné lieu à une publication scientifique présentée dans Baldanza *et al.* (2022a). Dans le chapitre suivant, nous présenterons une méthode qui base cette prédiction sur l'utilisation d'un réseau de neurones convolutif.

3.2 État de l'art

Dans la littérature, la problématique de suivi vidéo est associée au suivi d'objets. Cette tâche consiste à localiser et à suivre la trajectoire des objets tout au long de la vidéo. Comme expliqué dans la Section 1.2, on distingue deux types de suivi d'objets : le suivi

d'objet unique (*Single Object Tracking*) et le suivi d'objets multiples (*Multiple Object Tracking*). Pour répondre au contexte des vidéos sportives, nous nous concentrons ici principalement sur le suivi multi-objets.

3.2.1 Suivi d'objets

Marvasti-Zadeh *et al.* (2021) présente une étude globale et détaillée du sujet.

Bases de données

Comme pour beaucoup de tâches en vision par ordinateur, l'évaluation des méthodes de suivi d'objets peut se faire sur de nombreuses bases de données. Les principales sont : **MOT Challenge** (*Multiple Object Tracking*) (Leal-Taixé *et al.*, 2015), **KITTI** (Geiger *et al.*, 2013), **DukeMTMC** (*Duke Multi-Tracking Multi-Camera*) (Ristani *et al.*, 2016) ou encore **VOT** (*Visual Object Tracking*) (Kristan *et al.*, 2016).

Méthodes traditionnelles

Yilmaz *et al.* (2006) présente un des papiers les plus célèbres du domaine qui retrace une étude complète des méthodes de suivi d'objets dans les vidéos à cette époque. En 2006, les réseaux de neurones ne sont qu'à leurs prémices et les méthodes sont basées sur les caractéristiques des objets dans l'image. Les méthodes sont alors basées sur de nombreux principes mathématiques tels que la détection de points d'intérêt : Zhou *et al.* (2009) définit une méthode de suivi basée sur les **SIFT** (*Scale-Invariant Feature Transform*) pour relier les points d'intérêt entre les images et utilise le *mean shift* pour rechercher des similarités de couleurs. He *et al.* (2009) utilise lui les **SURF** (*Speeded Up Robust Features*), une extension des SIFT pour calculer le suivi à partir de l'extraction de points d'intérêt. Shin *et al.* (2005) présente une méthode qui base le suivi sur l'estimation du **flux optique** (*optical flow*). L'utilisation du flux optique est très fréquente pour cette tâche car elle permet d'estimer les mouvements de pixels entre deux images (autrement dit, elle permet d'estimer la nouvelle position d'un pixel entre une image et la suivante). Il existe beaucoup d'autres techniques, basées sur les couleurs (Nummiaro *et al.*, 2002), les formes (Wang et Yagi, 2008), les textures (Ning *et al.*, 2009),...

Comme dans beaucoup de tâches en vision par ordinateur, le développement des réseaux de neurones a complètement révolutionné le domaine. Notamment, la détection d'objets est devenue un des objectifs principaux dans le développement des réseaux de neurones.

Réseaux de neurones pour la détection d'objets

De nombreux concours et bases de données permettant d'évaluer les modèles sur cette tâche sont alors publiés tels que : **COCO** (*Common Object in Context*) (Lin *et al.*, 2014), **PASCAL VOC** (*Visual Object Classes*) (Everingham *et al.*, 2010), **ILSVRC** (*ImageNet Large Scale Visual Recognition Challenge*) (Russakovsky *et al.*, 2015), **KITTI** (Geiger *et al.*, 2013) et de nombreux autres.

Un des premiers réseaux de neurones convolutifs à avoir démontré une précision significative dans la détection d'objets (notamment sur PASCAL VOC et ILSVRC) est le **R-CNN** (*Regions with CNN features*) (Girshick *et al.*, 2014). Ce réseau propose des régions d'intérêt appelées ROIs (*region of interest*) dans l'image et les passe à travers un réseau de neurones convolutif pour extraire les caractéristiques. Ces caractéristiques sont ensuite utilisées pour classer les ROIs et pour prédire les boîtes englobantes des objets à l'intérieur. Les réseaux utilisés sont souvent des réseaux pré-entraînés tels que ceux présentés dans la Section 2.2 pour la classification : VGG-19 (*Visual Geometry Group*) ou ResNet. La Figure 3.7 présente un schéma du système présenté dans ce papier.

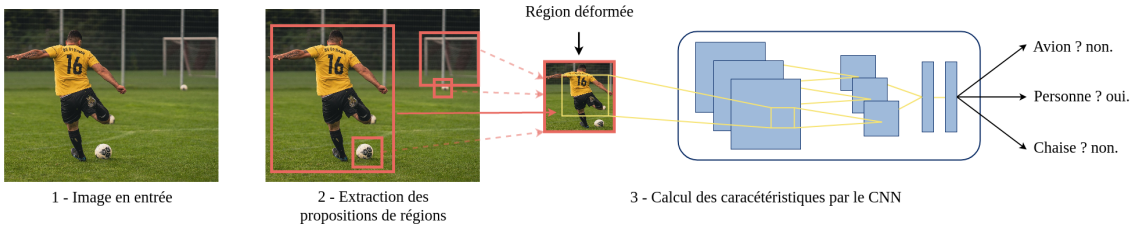


FIGURE 3.7 – **Schéma du système R-CNN pour la détection d'objets.** Ce schéma est inspiré de celui présenté dans Girshick *et al.* (2014). Le système prend en entrée une image (1) et extrait des propositions de régions (2). Chaque région passe ensuite dans un CNN pour extraire les caractéristiques (3). L'extraction des caractéristiques est utilisée pour classifier la région (4).

.

Une des limitations des R-CNN est leur temps de calcul relativement long, car chaque région proposée doit être passée individuellement à travers le réseau de neurones. Ces limitations ont donné lieu à des améliorations telles que le **Fast R-CNN** (Girshick, 2015) et le **Faster R-CNN** (Ren *et al.*, 2015). Ces architectures cherchent à unifier au maximum ces différentes tâches et à prédire une carte de caractéristiques unique au lieu de passer chaque ROI dans un réseau convolutif. Faster R-CNN utilise un réseau de neurones appelé RPN (*Region Proposal Network*) pour proposer automatiquement des ROIs à partir de la carte de caractéristiques commune. Le RPN utilise des ancres (*anchors*) pour proposer des régions qui pourraient contenir des objets, et ces régions sont ensuite classées et raffinées par le réseau de neurones pour la détection d'objets. Ce principe est d'ailleurs repris par des méthodes pour la proposition temporelle d'actions. D'autres méthodes visent à déve-

lopper des architectures plus rapides telles que **YOLO** (*You Only Look Once*) (Redmon *et al.*, 2016) et les **SSD** (*Single Shot MultiBox Detector*) (Liu *et al.*, 2016) qui prédisent les boites englobantes en une seule passe de réseaux de neurones. Plus récemment des architectures telles que **RetinaNet** (Lin *et al.*, 2017b) ou **EfficientDet** (Tan *et al.*, 2020) ont donné d'excellents résultats.

Suivi par détection

Comme expliqué dans l'introduction, il existe plusieurs types d'architectures pour le suivi d'objets. Une des plus connues consiste à détecter les objets dans les images et à les relier dans le temps. Cette technique s'appelle le suivi par détection. Bewley *et al.* (2016) présente **SORT** (*Simple Online and Real-time Tracking*), une méthode de suivi par détection basée sur l'utilisation du Faster R-CNN. Il présente une méthode de suivi multi-objets en temps réel (grâce aux performances des Faster R-CNN) qui détecte les objets dans l'image avant de les relier aux objets détectés dans les images précédentes. Pour cela, ils définissent un algorithme d'association qui utilise des critères tels que la distance ou la vitesse pour faire le lien entre les objets d'une image à l'autre ainsi qu'un algorithme permettant de prédire la future position d'un objet grâce aux filtres de Kalman (Kalman, 1960). Wojke *et al.* (2017) présentent **DeepSORT**, une extension de ce modèle qui extrait les caractéristiques visuelles des objets et les associe aux précédents grâce à un réseau de neurones convolutif. Bergmann *et al.* (2019) présentent **Tracktor**, un modèle de suivi uniquement basé sur la détection d'objets utilisant lui aussi les Faster R-CNN. Cet article présente un axe intéressant puisqu'il arrive à de très bons résultats en utilisant simplement un principe de suivi par détection et simple ré-identification des objets. Yu *et al.* (2016) présente une méthode qui utilise un Faster R-CNN modifié qui inclue notamment le *skip-pooling* (Bell *et al.*, 2016), qui consiste à ajouter des connexions directes pour sauter les couches de pooling. Cette méthode a permis d'obtenir des résultats de pointe sur la base de données MOT16 (Milan *et al.*, 2016). Zhang *et al.* (2019) définit un pipeline de suivi visant à suivre des porcs basé sur la détection d'objets. Dans ce papier, ils comparent notamment les résultats obtenus avec un SSD et un faster R-CNN. Kim *et al.* (2018) utilise lui l'architecture YOLOv2 (Redmon et Farhadi, 2017) pour réaliser la détection.

Réseaux siamois

Certaines études cherchent à s'affranchir de la détection d'objets pour réaliser le suivi. À partir des travaux de Bromley *et al.* (1993) qui définissent l'architecture siamoise (*Siamese Neural Network*) pour la vérification de signatures manuscrites, de nombreuses méthodes suivent les objets sans les détecter à chaque image. Ondrašovič et Tarábek (2021) présente une revue complète de cette stratégie pour le suivi d'objets vidéo. Les réseaux

siamois permettent de comparer deux entrées. Ils sont appelés ainsi car leur architecture est séparée en deux branches siamoises pour extraire les caractéristiques de chaque entrée. La sortie des branches est ensuite concaténée pour procéder à la comparaison et déduire l'analyse. La Figure 3.8 présente un schéma du fonctionnement de ce type d'architectures.

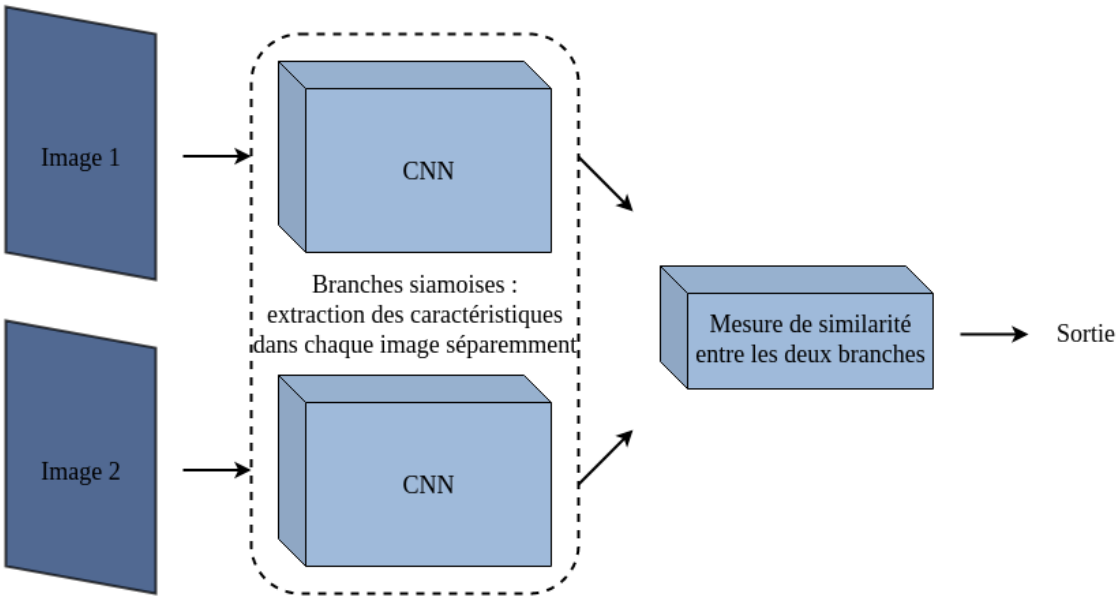


FIGURE 3.8 – **Schéma du fonctionnement d'une architecture siamoise.**

Pour le suivi, ces méthodes définissent une fonction de similarité qui permet de comparer deux patchs et en déduire s'ils sont composés du même objet ou non. Bertinetto *et al.* (2016) ont prouvé avec leur modèle **SiamFC** qu'il était possible d'obtenir des résultats compétitifs pour le suivi multi-objets avec des réseaux siamois. Held *et al.* (2016) présente **GOTURN** (*Generic Object Tracking Using Regression Networks*), une architecture siamoise qui utilise deux colonnes vertébrales pré-entraînées sur la base de données ImageNet (Deng *et al.*, 2009) pour profiter de leur capacité à extraire les caractéristiques dans les images. Beaucoup d'autres papiers présentent ensuite des méthodes basées sur l'architecture siamoise pour le suivi d'objets : Wang *et al.* (2017) présente **DCFNet** (*Discriminant Correlation Filters*), Valmadre *et al.* (2017) présente **CFNet** (*Correlation Filter)*, qui utilise une architecture siamoise asymétrique.

Réseaux de neurones récurrents

L'arrivée des réseaux de neurones récurrents (RNN), notamment le réseau LSTM (*Long Short-Term Memory*), ont permis des avancées significatives dans le suivi d'objets. Ces réseaux visent à traiter des séquences de données, en utilisant des boucles de rétroaction qui leur permettent de maintenir un état interne. Pour le suivi d'objets, le maintien de cet état interne permet au réseau d'utiliser des informations sur les images et les prédic-

tions précédentes pour calculer la prochaine prédiction. Milan *et al.* (2017) présente une méthode qui utilise deux RNN, dont un LSTM pour le suivi multi-objets vidéo. Le RNN prédit la position des cibles dans la nouvelle image et le LSTM permet de faire l'association avec les objets précédemment suivis. Sadeghian *et al.* (2017) utilise lui aussi les réseaux LSTM pour le suivi multi-objets. Ils présentent un modèle qui utilise un LSTM avec une couche *fully connected* (FC) pour fusionner les caractéristiques extraites par 3 autres LSTM. Ces réseaux visent à extraire les caractéristiques d'apparence, de mouvements et d'interactions. Certaines méthodes telles que Wan *et al.* (2018) combinent même les architectures siamoises avec l'utilisation des réseaux récurrents LSTM pour le suivi multi-objets.

3.2.2 Suivi d'objets dans les vidéos de sport

Le développement économique récent autour du traitement des vidéos sportives ainsi que les différentes problématiques que présente ce type de vidéos ont mené beaucoup d'études à présenter des méthodes de suivi d'objets dans les vidéos de sport. En effet les principales difficultés présentées dans les tâches de suivi vidéo (occlusions, changements de trajectoires) se multiplient dans le contexte sportif. Le ballon est souvent tenu ou caché par un joueur et les joueurs sont également souvent placés derrière d'autres. Les changements de trajectoires et les déplacements sont rapides et plus difficiles à prédire. Les méthodes de suivi classiques doivent généralement être adaptées pour être robustes dans ces conditions. Dans le contexte des sports collectifs qui nous intéressent, on distingue deux grands axes de problématiques. Les méthodes qui visent à détecter et à suivre le ballon et les méthodes qui visent à suivre les joueurs.

Suivi des joueurs

Avant 2010, les premières méthodes de suivi visant à suivre les joueurs adaptent les méthodes de suivi d'objets présentées à l'époque. Hu *et al.* (2010); Xing *et al.* (2010); Liu *et al.* (2009) présentent des méthodes pour suivre les joueurs qui reprennent les méthodes traditionnelles pour le suivi d'objets présentées dans la Section 3.2.1 pour les vidéos de sport. Dans Liu *et al.* (2009) et Xing *et al.* (2010), les auteurs définissent des méthodes de suivi par détection qui visent à localiser les joueurs dans l'image avant de les associer aux précédentes détections. Xing *et al.* (2010) utilise la méthode de détection de personnes présentée dans Viola et Jones (2001b) ainsi qu'une méthode de détection du terrain et de gestion des occlusions pour pouvoir traiter les problématiques du contexte sportif. La méthode présentée par Liu *et al.* (2009) utilise elle aussi un système de suivi par détection basé sur les travaux de Viola et Jones (2001a) pour détecter les personnes. Hu *et al.* (2010)

utilise une adaptation de la méthode de suivi *CamShift* présentée dans Allen *et al.* (2004) pour le suivi d'objets. Dans les papiers plus récents, les méthodes de suivi d'objets basées sur l'utilisation des réseaux ont également été adaptées pour répondre aux problématiques du suivi de joueurs. Buric *et al.* (2019) présente les performances d'algorithmes pour le suivi d'objets tels que SORT et DeepSort pour suivre les joueurs dans des vidéos de handball. Hurault *et al.* (2020) présente une méthode qui utilise un Faster R-CNN et un FPN (*Feature Pyramid Network*) (Lin *et al.*, 2017a) pour le suivi d'objets qu'ils ré-entraînent spécifiquement pour le suivi de joueurs dans des vidéos de football professionnel de leur base de données. Cette méthode permet notamment d'éliminer les détections de personnes n'étant pas des joueurs (entraîneurs, supporters, ...). Yoon *et al.* (2019) définit **Joy2019** une architecture adaptée de YOLO pour procéder au suivi par détection des joueurs dans leurs vidéos.

FIGURE 3.9 – **Exemple de reconnaissance de numéros pour le suivi de joueurs par la méthode Joy19.** Cette Figure est tirée de Yoon *et al.* (2019).

Dans ce papier, ils associent les joueurs détectés dans les nouvelles images aux joueurs précédemment suivis en ajoutant à la détection de personnes une reconnaissance des numéros portés par les joueurs détectés. Des exemples de la reconnaissance des numéros sur les maillots présentée par ce papier sont donnés dans la Figure 3.9.

Suivi du ballon

Comme précédemment, avant l'arrivée des réseaux de neurones, les méthodes de suivi du ballon utilisaient des méthodes traditionnelles de suivi basées sur l'analyse des caractéristiques présentées au-dessus. Ces méthodes devaient être adaptées pour pouvoir suivre un objet de petite taille tel que le ballon. Ohno *et al.* (1999) présente une méthode pour le suivi du ballon qui détecte les mouvements dans la zone radiale des joueurs dans plusieurs

images pour trouver la position du ballon. D'Orazio *et al.* (2004) utilise la CHT (*Circle Hough Transform*) pour détecter les formes circulaires dans une image et détecter la balle. Comme pour le suivi des joueurs, depuis le développement des réseaux de neurones, les méthodes de détection et de suivi d'objets sont adaptés pour la détection et le suivi du ballon. L'architecture **Joy19** présentée dans Yoon *et al.* (2019) permet lors de la détection des joueurs de déterminer si un joueur est en possession du ballon ou non. Pour les sports qui se jouent à la main, tels que le basket-ball et le handball, le ballon est régulièrement tenu par les joueurs donc difficilement détectable en tant que tel. Définir une façon de différencier le joueur en possession du ballon des autres donne sa position dans l'image. Grâce à ces détections, leur modèle est capable de suivre le ballon et de générer un graphe des relations de passes entre les joueurs. Cette architecture est testée sur des vidéos de basket-ball professionnel.

Kamble *et al.* (2019) présente **DLBT** (*Deep Learning Ball Tracking*) une autre méthode de suivi par détection qui utilise une architecture VGG-M inspirée de celles des VGG pour le suivi du ballon dans des vidéos de football professionnel. Cette méthode définit notamment un processus pour gérer les cas où le ballon est perdu par le tracker ou trop longtemps caché ainsi qu'un système d'IoU entre les images pour calculer le suivi.

Komorowski *et al.* (2019) présente **DeepBall**, une architecture inspirée de celles de YOLO et SSD pour la détection d'objet, adaptée pour détecter le ballon dans des images de football professionnel.

3.2.3 Positionnement de notre méthode

Beaucoup de méthodes de suivi d'objets en temps réel, et particulièrement dans le contexte de vidéos sportives, utilisent un pipeline de suivi par détection pour suivre les objets dans les vidéos. Si on souhaite utiliser une méthode de suivi d'objets pour diriger la caméra dans nos conditions, nous pourrions nous orienter vers la problématique de suivi du ballon. Le ballon est souvent assimilé au centre de l'action à filmer. Seulement, cette problématique présente des incompatibilités avec notre contexte. Premièrement, la forme et la position du ballon sont propres à chaque sport. Il faudrait donc définir une méthode par sport. Ensuite, les papiers que nous venons de présenter pour le suivi du ballon sont efficaces dans les conditions professionnelles auxquelles nous faisons référence dans l'introduction du chapitre. Cependant, dans le contexte amateur, les architectures utilisées pour la détection du ballon ne sont pas assez robustes. Burić *et al.* (2018) et les tests que nous avons fait sur nos données confirment les difficultés de ces méthodes à détecter et suivre le ballon de manière robuste dans le contexte du sport amateur. Nous avons testé deux architectures : YOLOv5 (Jocher *et al.*, 2022) et Faster R-CNN (Ren *et al.*, 2015) pré-entraînées sur la base de données COCO (Lin *et al.*, 2014), sur les vidéos courtes de

basket-ball et handball de la base de données de test présentée dans la Section 3.5. Les résultats obtenus sont présentés dans le Tableau 3.1. Ce tableau confirme que les architectures de détection d'objets ne parviennent pas à détecter le ballon de manière robuste sur nos images. Les deux modèles parviennent à détecter le ballon dans moins de 15% des images. Ces résultats sont trop peu fiables pour pouvoir être utilisés pour le suivi de manière robuste. De plus, les résultats obtenus par le Faster R-CNN ne sont pas calculés en temps réels. Pour pouvoir calculer cette détection en temps réel sur des processeurs embarqués, il faut sous-échantillonner les images. La détection d'objets de petite taille est moins performante sur des images sous-échantillonnées, ce qui rend le modèle moins précis. En temps réel, l'architecture Faster R-CNN parvient à détecter le ballon dans moins de 5% des images. Une autre possibilité serait de baser notre méthode sur le suivi des joueurs. Cependant, calculer le suivi précis et complet des joueurs dans une vidéo est une tâche coûteuse et difficilement réalisable en temps réel sur des systèmes embarqués à faible puissance de calcul.

% de ballons détectés	Basket-ball	Handball	Total
YOLOv5	2.12%	4.86%	3.2%
Faster R-CNN	10.3%	16.15%	13.4%

TABLEAU 3.1 – **Résultats des architectures pour la détection du ballon de la littérature sur nos images.** Ce tableau présente le pourcentage d'images de notre base de données dans lesquelles le ballon a été correctement détecté.

C'est pourquoi nous définissons dans ce papier une méthode qui permet de s'affranchir de la détection d'objets pour calculer le suivi d'action et la nouvelle position de la caméra. Notre méthode vise à simplifier l'analyse des déplacements des joueurs pour utiliser des opérateurs plus simples et pouvoir être calculés dans ces solutions. Enfin, notre approche permet de définir une méthode d'annotation automatique pour l'entraînement des modèles. Les données de la littérature pour notre étude étant inexistantes, nous devons construire notre base de données en intégralité. Une méthode permettant l'annotation automatique de celle-ci est un gain de temps précieux.

3.3 Flux optique et segmentation des plans

Dans cette thèse, nous présentons une méthode basée sur le flux optique pour prédire les déplacements de la caméra à partir des mouvements des joueurs. Pour cela, la méthode segmente le premier plan et l'arrière-plan du flux optique. Nous supposons ici le déplacement de l'arrière-plan correspondant au mouvement de la caméra. Nous souhaitons le distinguer des mouvements du premier plan qu'on considère comme les mouvements des

joueurs. L'estimation d'un flux optique dense (Liu, 2009) donne une matrice de vecteurs (u_i, v_i) résultant du déplacement des pixels dans l'image.

3.3.1 Segmentation premier plan/arrière-plan

Dans la suite de ce chapitre, on considère une image I comme un vecteur $\mathbb{R}^N$. Les joueurs sont définis comme le premier plan mobile de l'image. Dans nos images, on suppose que l'arrière-plan occupe plus de la moitié d'une image. Cela se justifie par le fait que les sports ne peuvent pas être capturés d'assez près pour que les joueurs occupent plus de 50 % de l'écran sans zoomer. On définit le mouvement global de l'arrière-plan au moment t comme la médiane 2D (u^*, v^*) du flux optique dense calculé entre les images t et $t-1$.

$$(u^*, v^*) = \underset{(u,v)\in\mathbb{R}^2}{\operatorname{argmin}} \sum_{i=1}^{N} ||(u, v) - (u_i, v_i)||. \tag{3.1}$$

avec (u_i, v_i) les valeurs issues du calcul du flux optique. (u^*, v^*) est trouvé en utilisant la méthode itérative des moindres carrés repondérés (Holland et Welsch, 1977).

On définit le flux optique du premier plan $f^t \in \mathbb{R}^{N\times 2}$ comme le vecteur de flux optique composé des éléments distants à un seuil θ du mode principal (u^*, v^*) défini à l'équation (3.1) et où les autres éléments sont fixés à 0. Pour rendre notre modèle robuste, il est important de dé-corréler les mouvements des joueurs calculés par le flux optique des mouvements de la caméra qu'on associe au mouvement global de l'arrière-plan (u^*, v^*). Pour cela, on ajoute (u^*, v^*) aux éléments non nuls du premier plan. Lors du calcul du flux optique, le mouvement d'un objet peut-être partiellement ou totalement compensé par le mouvement de la caméra. Ajouter ces valeurs aux éléments non nuls du flux optique segmenté permet d'annuler cette compensation. Ainsi, le vecteur f^t a N composantes f_i^t définies comme suit

$$f_i^t = \begin{cases} (u_i + u^*, v_i + v^*), & \text{si } ||(u_i, v_i) - (u^*, v^*)||^2 \geq \theta. \\ 0 & \text{sinon .} \end{cases} \tag{3.2}$$

Comme le seuil θ doit être adapté pour chaque image, nous proposons d'utiliser

$$\theta = \lambda \frac{\sum_{i=1}^{N} ||(u_i, v_i) - (u^*, v^*)||^2}{N} \tag{3.3}$$

avec λ une constante définie expérimentalement pour maximiser la qualité de la segmentation. Plus cette valeur est grande, plus les valeurs du flux optique doivent être éloignées de (u^*, v^*) pour être conservées au premier plan.

FIGURE 3.10 – **Exemples de segmentations réalisées grâce au flux optique.** À gauche, des images prises dans une vidéo. Au milieu, le flux optique calculé à partir de l'image de gauche et de l'image précédente. À droite, la segmentation premier plan/arrière-plan obtenue.

La Figure 3.10 montre des exemples de segmentation premier plan/arrière-plan obtenus avec cette méthode. Dans la suite, nous ne considérons que la composante horizontale du flux optique. Les déplacements de la caméra pour le basket-ball et le handball utilisés dans nos exemples sont essentiellement horizontaux. Pour d'autres sports, la composante verticale peut facilement être ajoutée au modèle.

3.3.2 Normalisation

Cette partie présente une méthode de normalisation du flux optique. Ici, une normalisation des données est obligatoire pour augmenter la robustesse du modèle et pour traiter les fortes variations entre les positions d'acquisition induites par le contexte amateur (position du caméraman dans les tribunes, distance au terrain ...). La position latérale de la caméra est supposée connue car la caméra doit être placée au milieu du terrain pour rendre notre modèle cohérent.

Echelle

Tout d'abord, en raison du contexte amateur, le modèle doit être robuste aux grands changements de taille des joueurs induits par les variations de la distance entre la position de la caméra et le terrain. Pour éviter l'impact de ces variations, nous définissons n_f le nombre de pixels activés au premier plan et n_b ceux activés à l'arrière-plan. Nous définissons le facteur d'échelle $\eta \in \mathbb{R}$ comme suit

$$\eta = \left(\frac{n_b}{n_f}\right)^{\alpha}, \tag{3.4}$$

où $\alpha \in \mathbb{R}$ est un paramètre utilisé pour ajuster l'impact de la normalisation sur la prédiction (fixé à 1.3 dans nos expériences). La normalisation de l'échelle est définie comme ci-dessus car le modèle de prédiction défini dans l'équation (3.5) dépend du nombre de pixels activés au premier plan. Si la caméra est placée loin du champ, nous obtiendrons des valeurs élevées pour le facteur d'échelle η et inversement.

Hauteur de la caméra et orientation

Comme les images sont prises à partir d'un seul endroit, nous ne pouvons pas normaliser la position de la caméra en utilisant une homographie complète. Dans ces conditions, nous utilisons une transformation linéaire directe (Dubrofsky, 2009) pour la normaliser. Notre modèle utilise un redimensionnement linéaire pour normaliser l'impact de la position verticale et de l'orientation de la caméra sur l'apparence des joueurs. Comme expliqué dans la partie sur la normalisation de l'échelle, notre modèle de prédiction dépend du nombre de pixels activés au premier plan, et l'orientation de la caméra peut avoir un fort impact sur la taille des joueurs dans l'image. La profondeur est normalisée par η. Le redimensionnement linéaire est appliqué au résultat de la segmentation f^t. Pour déterminer le rescaling pour chaque f^t, nous appliquons une correspondance verticale de 2 points entre le flux optique du premier plan et l'espace normalisé. On utilise pour cela les points qui permettent d'englober verticalement 90% des valeurs. Nous laissons 5% de marge en haut et en bas pour être robuste aux erreurs de segmentation. La Figure 3.11 montre des exemples de résultats de normalisation de l'environnement sur la position du premier plan. Nous voyons que les joueurs sont replacés verticalement et que l'envergure verticale est ajustée.

3.4 Modèle de prédiction linéaire par morceaux

Notre modèle vise à déduire le mouvement de la caméra nécessaire au suivi de l'action à partir des mouvements des joueurs calculés dans (3.2). Cette partie présente la façon dont le modèle déduit la direction globale de l'action à partir du flux optique segmenté. L'idée est de transformer une localisation d'action complexe en un simple problème linéaire par morceaux. Le coût computationnel de la prédiction linéaire permet de suivre l'action en temps réel.

3.4.1 Modèle linéaire

Soit $d^t \in \mathbb{R}$ le déplacement de la caméra que nous voulons déduire au temps t de f^t construit comme dans l'équation (3.2). Nous pouvons résoudre le problème en définissant

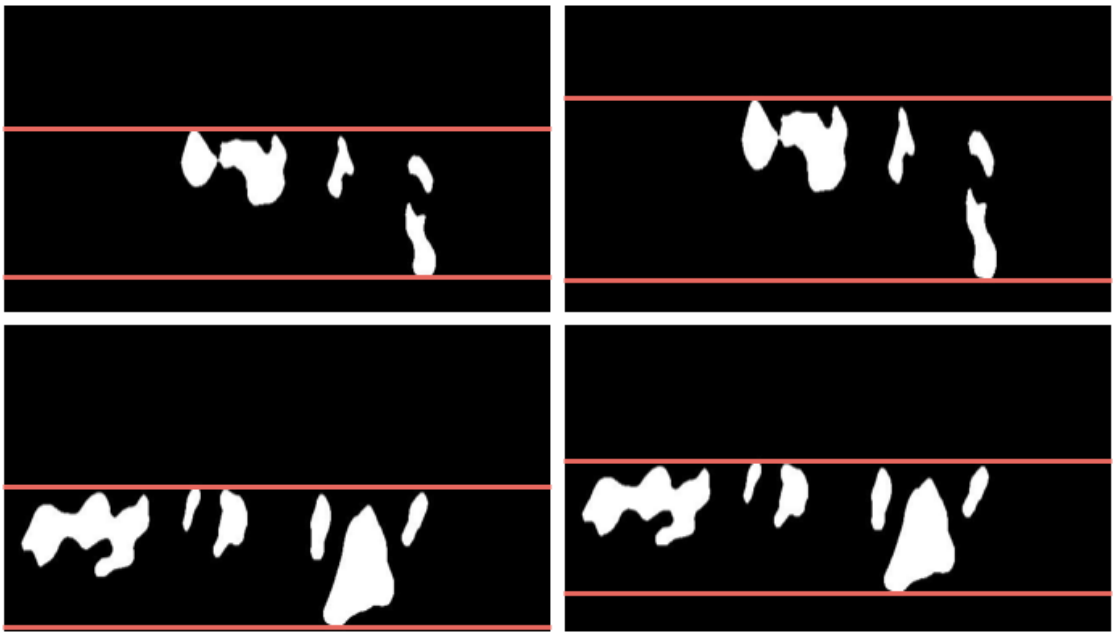


FIGURE 3.11 – **Impact de la normalisation sur les segmentations.** À gauche, segmentation premier plan/arrière-plan avant la normalisation. À droite, segmentation premier plan/arrière-plan après le redimensionnement linéaire.

le modèle linéaire

$$d^t = \langle f^t \eta, z \rangle + e^t \tag{3.5}$$

où $z \in \mathbb{R}^N$ est un vecteur de poids appris, $e^t \in \mathbb{R}$ l'erreur de prédiction et η le paramètre de normalisation défini dans (3.4).

Au cours d'une vidéo, les mêmes mouvements de joueurs peuvent induire des mouvements de caméra différents selon la situation. Par exemple, tous les joueurs qui courent dans une direction nécessiteront un mouvement de caméra si celle-ci est située à une extrémité du terrain et pas à l'autre. En conséquence, notre modèle doit être capable de prédire différents mouvements en fonction de la situation pour une même segmentation.

3.4.2 Modèle linéaire par morceaux

On note s une situation associée à un type d'action de jeu et on étend le modèle (3.5) au modèle linéaire par morceaux

$$d^t = \langle f^t \eta, z_s \rangle + e^t \tag{3.6}$$

où z_s est un vecteur de poids différent appris pour chaque situation différente. Sur une caméra PTZ, la position de la caméra est supposée connue en raison de la calibration. La gestion des situations peut être déduite à partir de la direction de la caméra. Pour éva-

luer les prédictions sur nos données, la gestion des situations est donc faite en analysant l'évolution des valeurs de la vérité terrain. L'idée derrière la séparation des situations est de rendre le modèle adaptatif à un maximum de comportements différents. En définissant des situations adaptées, ce modèle est extensible à d'autres sports et pas seulement au basket-ball.

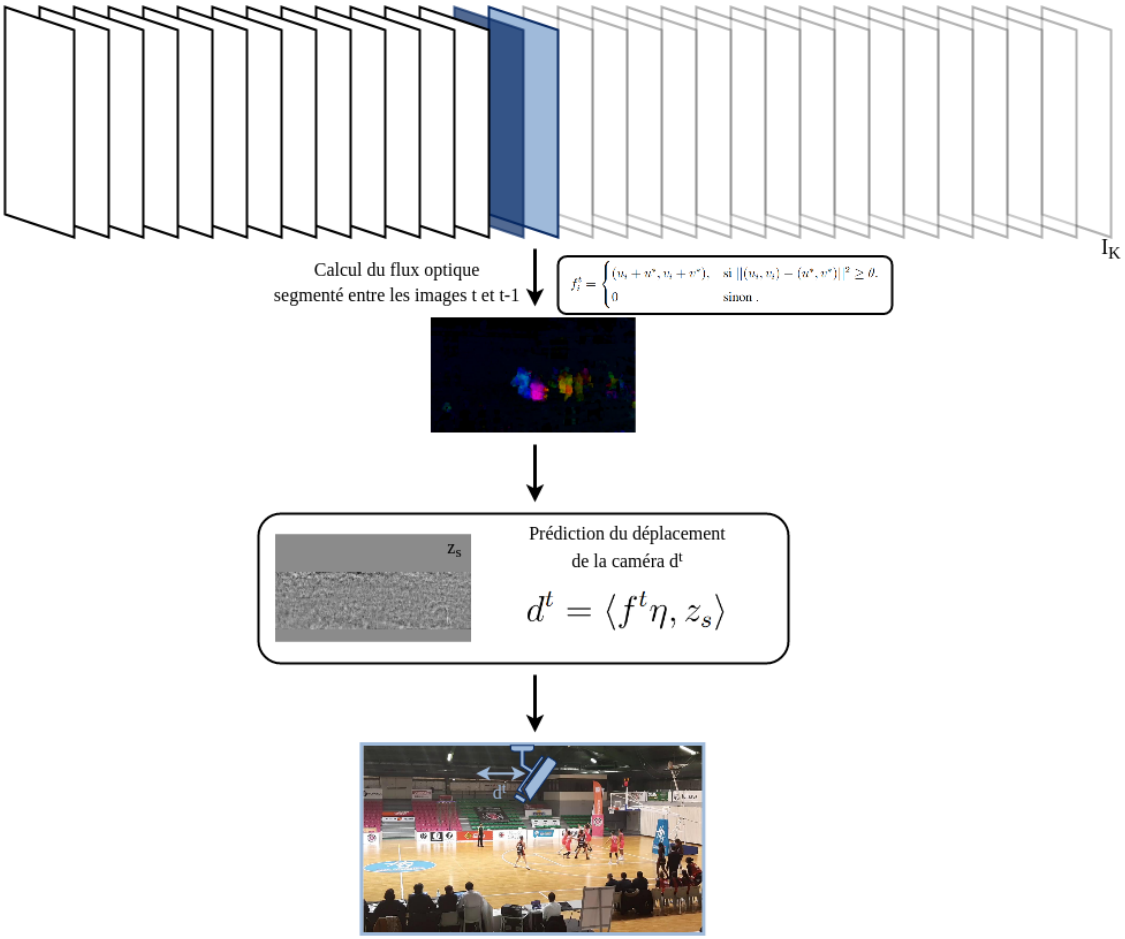


FIGURE 3.12 – **Schéma du pipeline du modèle de prédiction linéaire par morceaux.**

Pour améliorer les performances et limiter les erreurs, chaque prédiction est lissée avec un noyau gaussien et seuillée pour forcer les petites valeurs à 0 afin d'éviter que les résultats ne dérivent.

3.4.3 Apprentissage et situations

Pour ce modèle, les bases de données d'apprentissage sont divisées en deux situations : les actions commençant du côté gauche du terrain ($s = 0$) et celles du côté droit ($s = 1$). Nous entraînons donc 2 matrices z_r et z_l par apprentissage supervisé pour chaque situation en utilisant l'optimiseur ADAM (Kingma et Ba, 2014). La situation de départ est annotée manuellement et les changements de situation sont gérés en analysant la somme

des déplacements réels. La vérité terrain étant normalisée en pourcentage du terrain, on change de situation lorsque la somme des déplacements atteint 100%. Cela revient à utiliser une information sur la direction de la caméra pour gérer les situations. Pour apprendre le modèle, la fonction de Loss est définie comme suit

$$L(z_s) = ||F_s z_s - d_s||^2 + \frac{1}{\epsilon}\sum_{i=1}^{K}\Phi_i(z_s) \tag{3.7}$$

où K est la taille du batch, F_s est une matrice composée de K vecteurs de flux optique segmentés et normalisés, d_s est un vecteur composé des K labels associés au mouvement de la caméra. Φ_i sont des fonctions de pénalisation qui favorisent une direction de mouvement correspondant à chaque situation (mouvement vers la droite pour $s = 0$ et vers la gauche pour $s = 1$ typiques dans ces sports).

$$\Phi_i(z) = max((-1)^{s+1}\langle F_s^i, z_s\rangle, 0)^2. \tag{3.8}$$

Le taux d'apprentissage est fixé à $\alpha = 10^{-6}$. Le paramètre ϵ est fixé à 0.1. Il faut noter que la pénalisation Φ doit être adaptée aux sports considérés avec une connaissance à priori sur chaque situation donnée.

3.5 Bases de données et annotations

3.5.1 Bases de données

Pour montrer la capacité de notre méthode à s'adapter à plusieurs sports, ce travail est réalisé sur des vidéos de basket-ball et le handball. Les deux sports nécessitent cependant des comportements différents en fonction des déplacements des joueurs. Nous avons donc utilisé deux bases de données d'entraînement séparées (60494 flux optiques pour le basket-ball et 42654 pour le handball) provenant de 162 vidéos prises sur la plateforme Rematch. Les vidéos de ces bases de données ont été pré-sélectionnées pour que la caméra soit placée proche du milieu du terrain et que le suivi de l'action soit précis et fluide. Pour les tests, les performances sont évaluées sur 3 jeux de données différents. Nous disposons d'un jeu de vidéos de basket-ball courtes (10 secondes), un jeu de vidéos de basket-ball longues (allant de 24 secondes à plus de deux minutes) et un jeu de vidéos courtes de handball (12 secondes). Tester les méthodes sur des vidéos longues permet de mettre en avant la précision des méthodes au cours du temps.

Augmentation des données

Les terrains de sport étant symétriques, toute image des bases de données d'apprentissage décrites ci-dessus peut être dupliquée et inversée pour obtenir la version miroir. Les images augmentées sont obtenues en opérant une symétrie axiale le long de l'axe vertical pris au milieu de l'image. Les vecteurs de flux optique et les labels associés sont remplacés par leur opposé. Comme ce processus peut être effectué sur chaque image, cette augmentation des données permet de multiplier par deux l'ensemble des données d'entraînement.

3.5.2 Annotation

Le principal avantage de ce travail est que les bases de données peuvent être annotées automatiquement. Le déplacement de la caméra peut être considéré comme la médiane 2D du flux optique définit à l'équation (3.1) et supposé être le mouvement global de l'arrière-plan calculé au temps t. Le mouvement à prédire est annoté avec cette valeur, normalisée en pourcentage du terrain. La cohérence des annotations a été vérifiée manuellement pour chaque vidéo. La comparaison entre les courbes obtenues avec cette méthode et le contenu des vidéos nous permet de montrer que le mouvement global calculé de l'arrière-plan, défini comme (u^*, v^*) , est bien associé au comportement de la caméra. En supposant que la somme de ces déplacements représente 100 % du terrain, on normalise chaque mouvement en pourcentage du terrain. Cette normalisation permet de réduire la subjectivité et la variabilité des labels entre les vidéos. Elle permet également au modèle d'être robuste aux changements d'échelle pendant l'apprentissage, car ces variations peuvent induire des déplacements d'arrière-plan différents pour le même mouvement de caméra.

3.6 Résultats

Dans cette partie, nous évaluons les performances du modèle linéaire par morceaux pour le suivi d'action vidéo sur nos données. L'objectif de notre méthode étant de prédire les déplacements de la caméra, l'annotation des données rend impossible la comparaison de nos résultats avec les méthodes de suivi d'objets de la littérature telles qu'elles sont présentées. Comme mentionné dans l'introduction de ce chapitre, nous avons évalué des méthodes de détection d'objets de l'état de l'art pour localiser le ballon dans nos images. Les performances obtenues sont trop faibles pour pouvoir suivre l'action sur les vidéos de notre base de données. Ces modèles obtiennent des résultats trop peu précis pour être analysés dans cette partie. Les tests de notre modèle ont été réalisés en deux temps. Dans un premier temps, pour valider l'approche et certains choix, le modèle a été entraîné

sur une sous-base de données d'apprentissage constituée de 4050 flux optiques issus de 27 vidéos de basket-ball. Ainsi, nous avons pu évaluer les choix de pénalisation et de traitement des prédictions (lissage et seuillage) sur les 15 vidéos courtes de basket-ball. Ensuite, nous évaluons le modèle entraîné sur la base de données complète présentée dans la Section 3.5. On présente alors les résultats du modèle sur toute la base de données de test.

3.6.1 Métrique d'évaluation

Comme on l'explique dans la Section 3.1, il est compliqué d'évaluer les performances d'une méthode pour le contrôle autonome de caméra de manière hors ligne. Le mouvement de la caméra prédit induisant la capture de l'image suivante, chaque prédiction a une influence sur la précédente qu'il est compliqué de reproduire en testant le modèle de cette façon. Il faut donc prendre ce paramètre en considération au moment de l'évaluation. Pour évaluer les résultats du modèle, nous utilisons l'erreur absolue moyenne, appelée **MAE** (*Mean Absolute Error*), sur les prédictions.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{3.9}$$

avec n le nombre de prédictions dans la vidéo, y_i la valeur de la prédiction et x_i la vérité terrain.

Cette analyse permet d'évaluer localement la capacité du modèle à prédire des mouvements proches des déplacements réels de la caméra. Nous évaluons également la MAE sur les prédictions intégrées en fonction du temps. Bien que nous ne puissions pas évaluer les prédictions sur l'image suivante issue du déplacement prédit, évaluer l'intégration en fonction du temps des prédictions permet d'analyser la capacité du modèle à reproduire le comportement d'acquisition humain au cours du temps. Évaluer l'intégration des prédictions au cours du temps permet de connaître la position de la caméra prédite par rapport à sa position au début de la vidéo et de la comparer avec le déplacement de la caméra réelle. On appellera cette métrique l'**erreur de suivi** TE (*Tracking Error*). Dans notre base de données, une image contient entre 40% et 60% du terrain en fonction de la distance à laquelle est placée la caméra. Pour enlever toute subjectivité, on doit considérer en théorie que le modèle ne parvient pas à reproduire l'acquisition réelle si la position de la caméra prédite s'éloigne de plus de 20% du centre de l'action réel (i.e. le centre de l'image capturée). Pour être plus robuste dans notre évaluation, on considérera qu'une vidéo est mal suivie si son erreur de suivi est supérieure à 15%. Sur les vidéos longues, pour s'assurer de la qualité du suivi sur toute la vidéo, il faut également analyser l'erreur de suivi loca-

	MAE (% du terrain)			Tracking Error (% du terrain)		
Vidéo	NO PEN	**PEN**	NO PP	NO PEN	**PEN**	NO PP
1	**0.01**	**0.01**	0.22	1.03	**0.10**	2.21
2	0.27	**0.23**	0.42	**4.17**	5.67	12.33
3	**0.72**	0.86	1.09	**10.26**	13.70	22.05
4	0.62	**0.58**	0.95	14.02	**9.85**	23.9
5	**0.32**	0.33	0.46	15.27	**11.41**	14.77
6	**0.22**	0.25	0.52	**5.69**	12.36	11.6
7	**0.45**	0.55	0.70	**4.71**	4.72	5.91
8	0.58	**0.36**	0.61	**4.84**	4.99	6.47
9	**0**	0.11	0.58	**0**	3.58	24.12
10	0.72	**0.54**	0.82	12.05	**6.26**	13.10
11	**0**	**0**	0.15	**0**	**0**	2.70
12	0.66	**0.65**	0.97	15.74	**12**	17.55
13	0.98	**0.86**	1.07	24.58	**18.57**	23.33
14	0.27	**0.11**	0.25	2.92	4.18	**2.41**
15	**0.35**	0.41	0.57	8.40	**7.01**	9.97
MEAN	0,41	**0.39**	0.63	9.29	**7.62**	12.23
BEST	8	**9**	0	7	**8**	1
			MAX	24.58	**18.57**	24.12
			NUMBER OF TE >15	3	**1**	4

TABLEAU 3.2 – **Résultats préliminaires du modèle sur la base de données des vidéos courtes de basket-ball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) avec les trois versions du modèle évalué pour chaque vidéo de la base de données. Le tableau montre que PEN est le meilleur modèle sur cette base de données et donne de bons résultats de suivi sur 14 vidéos (Tracking Error $\leq$ 15%).

lement. Pour savoir si une vidéo est mal suivie dans le temps, on regarde l'erreur de suivi maximale sur des sous-séquences de 10 secondes de la vidéo. On appelle cette métrique MAX TE (*Maximum Tracking Error*). Si cette valeur est supérieure à 15% du terrain, on considère que le modèle n'a pas réussi à reproduire l'acquisition réelle sur une partie de la vidéo.

3.6.2 Évaluation

Résultats préliminaires

Pour valider notre approche et les choix présentés dans la méthode, nous avons évalué notre modèle sur une base de données de 15 vidéos courtes de basket-ball. Les résultats sont présentés dans le Tableau 3.2.

Dans ce tableau, on compare une version de la méthode entraînée avec les fonctions

de pénalisation dans la Loss d'apprentissage (PEN), une version du modèle entraînée sans ces fonctions (NO PEN) et une version de la méthode apprise avec les fonctions de pénalisation mais qui n'utilise pas le processus de lissage et seuillage des prédictions (NO PP).

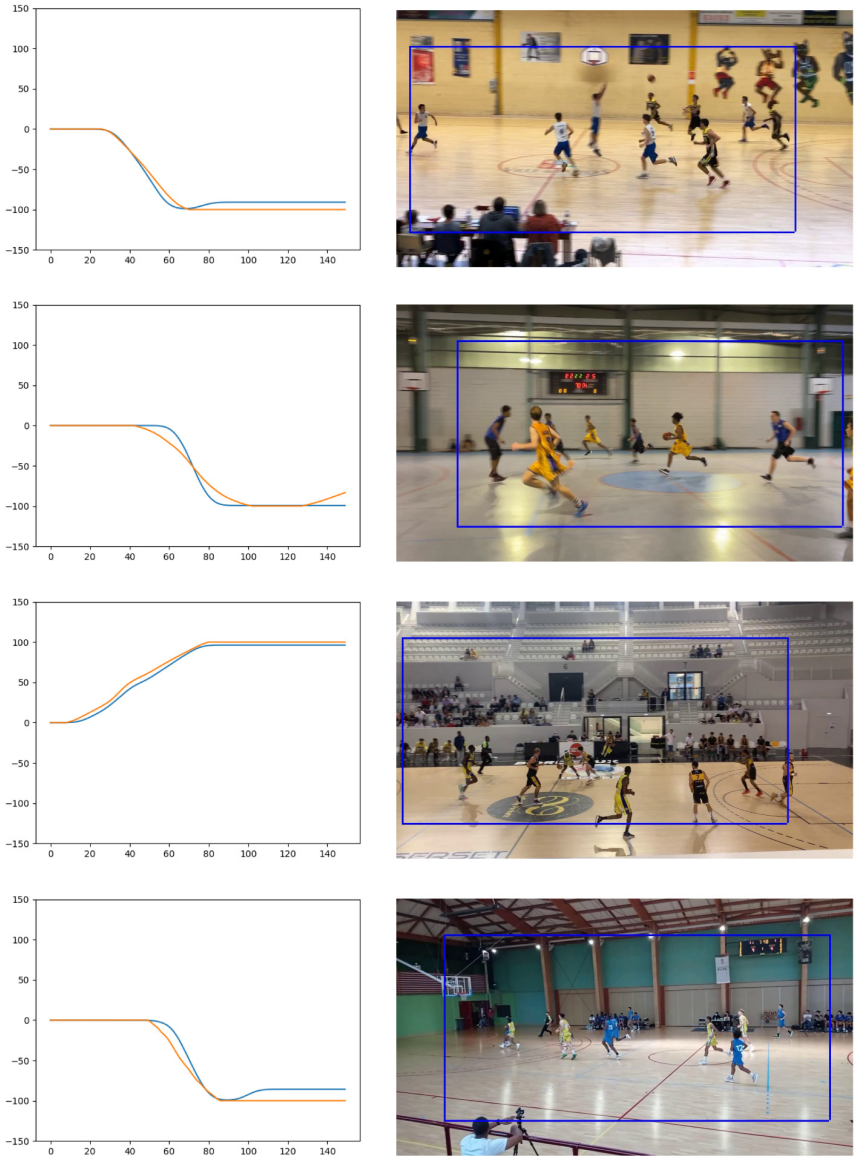


FIGURE 3.13 – **Exemples des résultats du suivi sur les vidéos n°2, 7, 14 et 15 de la base de données.** La courbe orange correspond à la vérité terrain et la courbe bleue correspond aux prédictions. Dans les images, le rectangle bleu schématise la position de la caméra prédite et permet de mettre en avant le décalage entre la caméra prédite et l'image réelle.

Les résultats présentés dans ce tableau montrent que PEN est la version la plus performante et permettent de justifier de l'utilisation des fonctions de pénalisation pendant l'apprentissage ainsi que du processus de lissage et seuillage des prédictions. Notamment

lors des phases de jeu où la caméra est fixe, ce processus permet de limiter les erreurs et le phénomène de dérive. Le modèle PEN obtient les meilleurs résultats sur tous les plans. Lorsqu'on évalue la précision locale des prédictions, il obtient la plus faible MAE moyenne sur la base de données ainsi que sur le plus grand nombre de vidéos. Cela signifie que ce modèle est celui qui parvient à prédire les mouvements de caméra les plus proches des mouvements réels. C'est donc le modèle qui parvient à imiter au mieux le comportement d'acquisition humain. Lorsqu'on regarde les prédictions intégrées en fonction du temps, ce modèle est également celui qui obtient les meilleurs résultats. Il obtient une TE supérieure 15% dans une seule vidéo. Observer ce type de résultat valide notre approche quant à sa capacité à apprendre et à reproduire l'acquisition vidéo humaine à partir de l'analyse des mouvements des joueurs dans l'image. Étudier les prédictions intégrées en fonction du temps nous permet également d'observer les courbes de déplacement de la caméra qui nous permettent de mieux comprendre les résultats obtenus. L'observation de ces courbes dans la Figure 3.13 permet de valider visuellement notre approche pour reproduire l'acquisition réelle. Sur ces vidéos, on voit que les prédictions sont très proches de la vérité terrain tout au long de la vidéo.

Vidéos courtes

Dans cette partie, on évalue le modèle entraîné sur toute la base de données présentée dans la Section 3.5, ce qui explique que les résultats soient différents de la section précédente. Le Tableau 3.3 présente les résultats du modèle sur les vidéos courtes des bases de données de handball et de basket-ball.

Ces résultats montrent que le modèle est capable de prédire localement des mouvements de caméra très proches des mouvements de caméra réels. Notamment, entraîner le modèle sur la base de données complète a permis d'obtenir des prédictions beaucoup plus proches de la vérité terrain que lors de l'évaluation préliminaire (la MAE moyenne sur les 15 vidéos de basket-ball est passée de 0.39 à 0.18). Ces résultats montrent également la capacité du modèle à fonctionner dans des sports différents. Les résultats évalués sur les vidéos de basket-ball et de handball sont très proches. Ces résultats viennent consolider l'intuition derrière notre approche. Notamment sur la base de données en handball, le modèle présente d'excellents résultats puisqu'il est parvenu à reproduire les déplacements de la caméra réelle au cours du temps sur toutes les vidéos. Cependant, on distingue que sur 3 vidéos de la base de données de basket-ball, les déplacements n'ont pas été correctement reproduits par la méthode. Ces vidéos étant correctement traitées dans les résultats préliminaires, on peut en déduire que ce résultat est dû à un manque de flexibilité du modèle qui peut présenter des difficultés avec l'apprentissage de données trop variées.

%	Basket-ball		Handball	
Vidéo	MAE	Tracking Error	MAE	Tracking Error
1	0.0001	0.05	0.21	4.92
2	0.16	2.00	0.31	10.68
3	0.3	9.12	0.10	14.04
4	0.39	**24.72**	0.1	4.02
5	0.25	**36.62**	0.19	9.27
6	0.27	**37.92**	0.15	4.82
7	0.19	13.01	0.1	5.14
8	0.12	2.78	0.26	9.36
9	0.0001	0.04	0.14	2.97
10	0.27	4.99	0.15	6.17
11	0.0001	0.14	0.24	9.65
12	0.28	6.66	0.22	7.02
13	0.26	14.96	0.14	4.1
14	0.10	14.89	0.15	4.80
15	0.14	12.65	-	-
MEAN	0.18	12.04	0.18	6.93
NUMBER OF TE >15		**3**		**0**

TABLEAU 3.3 – **Résultats du modèle linéaire par morceaux sur les bases de données des vidéos courtes de basket-ball et handball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) obtenues par le modèle **PEN** sur chaque vidéo de ces bases de données. Le modèle parvient à reproduire l'acquisition réelle dans 26 vidéos sur 29.

Vidéo	Durée	MAE	Tracking Error	MAX TE
1	0 :34	0.15	6.67	10.69
2	0 :24	0.001	0.05	0.057
3	2 :18	0.1	14.14	**45.86**
4	0 :30	0.001	0.11	0.12
5	1 :42	0.12	**23.08**	**39.50**
6	0 :53	0.16	11.21	**17.72**
7	0 :56	0.14	**24.15**	**53.82**
MEAN		0,1	11.34	-

TABLEAU 3.4 – **Résultats du modèle linéaire par morceaux sur les bases de données des vidéos longues de basket-ball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) obtenues par le modèle **PEN** sur chaque vidéo de cette base de données. Le modèle ne parvient pas à reproduire l'acquisition réelle dans 4 vidéos de cette base. Il présente des difficultés à suivre des longues séquences.

Vidéos longues

Le Tableau 3.4 présente les résultats sur la base de données des vidéos longues. Localement, les modèles obtiennent là aussi de très bons résultats. Cependant, lorsqu'on évalue les performances du modèle au cours du temps, on constate qu'il obtient de mauvaises performances de suivi (MAX TE > 15 dans 4 vidéos et Tracking Error > 15 dans 2 vidéos). Ces résultats mettent en avant les lacunes de notre modèle. En effet sur des vidéos plus longues, la gestion des changements de situation peut générer des erreurs dans le suivi de l'action qui font perdre le fil du match au modèle. Les résultats présentés dans cette partie permettent de valider l'intuition derrière l'approche, cependant, ils mettent en avant certaines difficultés à réaliser le suivi sur des longues séquences de jeu.

3.7 Conclusion du chapitre

Dans ce chapitre, nous avons présenté une nouvelle approche pour le suivi vidéo d'action dans le sport amateur visant à permettre le contrôle autonome de caméra PTZ dans ces conditions. Pour pallier les problématiques amenées par le contexte amateur qui rendent inutilisables les méthodes de suivi basées sur la détection d'objets, nous avons conçu une méthode qui base ses prédictions sur la segmentation des mouvements des joueurs dans les images. Ces mouvements sont calculés par l'estimation du flux optique. Ainsi, notre méthode utilise un modèle d'apprentissage supervisé linéaire par morceaux pour déduire directement de ces mouvements les déplacements de la caméra nécessaires pour suivre l'action. Les résultats obtenus ont montré des performances très encourageantes qui va-

lident notre approche. Ces tests ont aussi permis de mettre en avant des faiblesses dans notre méthode, notamment la flexibilité du modèle à apprendre des données très variables ainsi que pour suivre des longues séquences de match. Pour pallier ces difficultés, nous allons définir dans le prochain chapitre une nouvelle méthode qui reprend les bases de celle-ci, mais qui calcule cette fois la prédiction du mouvement de la caméra grâce à un réseau de neurones convolutif. En effet, l'utilisation de réseaux profonds devrait permettre de s'affranchir de la gestion des situations qui pose problème dans le suivi des séquences longues ainsi que d'ajouter des paramètres pour permettre l'apprentissage de données plus variées.

Chapitre 4

Suivi d'action pour caméra autonome : réseau de neurones convolutif

Sommaire

4.1 Introduction 79
4.1.1 Contexte 79
4.1.2 Motivations 80
4.2 Architecture CNN-3D pour le suivi d'action : NetCamMot 81
4.2.1 Reformulation 81
4.2.2 Détails de l'architecture 82
4.3 Base de données de vidéos fisheye pour l'évaluation 84
4.3.1 Vidéos 84
4.3.2 Annotation 85
4.4 Résultats 85
4.4.1 Métrique d'évaluation 86
4.4.2 Comparaison avec le modèle linéaire par morceaux 86
4.4.3 Caméra virtuelle 91
4.5 Conclusion du chapitre 95

4.1 Introduction

4.1.1 Contexte

Dans ce chapitre, nous reprenons l'approche présentée dans le chapitre précédent en remplaçant le modèle linéaire par morceaux par un **réseau de neurones convolutif**. L'évaluation des performances dans le chapitre précédent a montré des résultats encourageants

pour notre approche. Les expériences ont montré que baser le suivi sur la prédiction en une étape des mouvements de la caméra à partir des déplacements des joueurs dans les images permet de reproduire l'acquisition humaine de façon précise. Ici, nous allons présenter une nouvelle méthode qui vise à améliorer les performances obtenues précédemment. La méthode présentée dans ce chapitre fonctionne comme la précédente. Elle prend en entrée une vidéo de sport amateur et cherche à prédire le mouvement d^t de la caméra nécessaire pour suivre l'action à l'instant t à partir des images précédentes. La prédiction est ici aussi calculée à partir des segmentations du premier plan du flux optique.

Pour discuter des améliorations amenées par l'utilisation d'un réseau de neurones, nous comparerons la méthode avec celle du chapitre précédent sur la base de données présentée dans la Section 3.5. Ensuite, pour évaluer les performances en conditions réelles et pour que chaque prédiction influe sur l'image prochainement capturée, on évaluera la méthode sur des vidéos de sport amateur capturées avec une **caméra fisheye**.

Definition 4.1.1. Une caméra à objectif fisheye, généralement appelée caméra fisheye est une caméra munie d'un objectif avec une **distance focale très courte**. Cette particularité leur permet de capturer un champ de vision plus large que les caméras standard. L'angle de champ de certaines caméras fisheye peut dépasser 180°.

Grâce à ce type de caméra, l'intégralité du terrain peut être capturée. On considérera alors une portion d'image fisheye comme l'image capturée par une caméra PTZ classique. C'est cette portion d'image qui sera passée en entrée du modèle pour prédire les déplacements nécessaires au suivi de l'action. Ainsi à chaque instant t, on peut extraire la portion d'image correspondant à la prédiction précédente pour calculer la prédiction suivante. Ce processus nous permet de définir une caméra PTZ virtuelle, et d'évaluer les performances du modèle comme si on utilisait une vraie caméra PTZ. Ce processus est schématisé dans la Figure 4.1. Les tests sur des vidéos déjà capturées nous permettent d'analyser la capacité du modèle à reproduire une acquisition déjà réalisée, mais pas d'évaluer la capacité réelle à suivre l'action. Dans la réalité, une erreur de prédiction a des conséquences beaucoup plus lourdes sur la suite des résultats. Il est donc essentiel d'évaluer notre modèle en conditions réelles pour montrer qu'il est performant.

4.1.2 Motivations

Bien que le modèle linéaire par morceaux obtienne des résultats intéressants, il montre aussi certaines limites. Comme expliqué précédemment, cette méthode a montré des difficultés à suivre des longues séquences de jeu ainsi qu'à apprendre des données plus larges et variées. Pour répondre à ces limites, nous allons définir une méthode basée sur une architecture de réseau de neurones convolutif pour prédire le déplacement de la caméra.

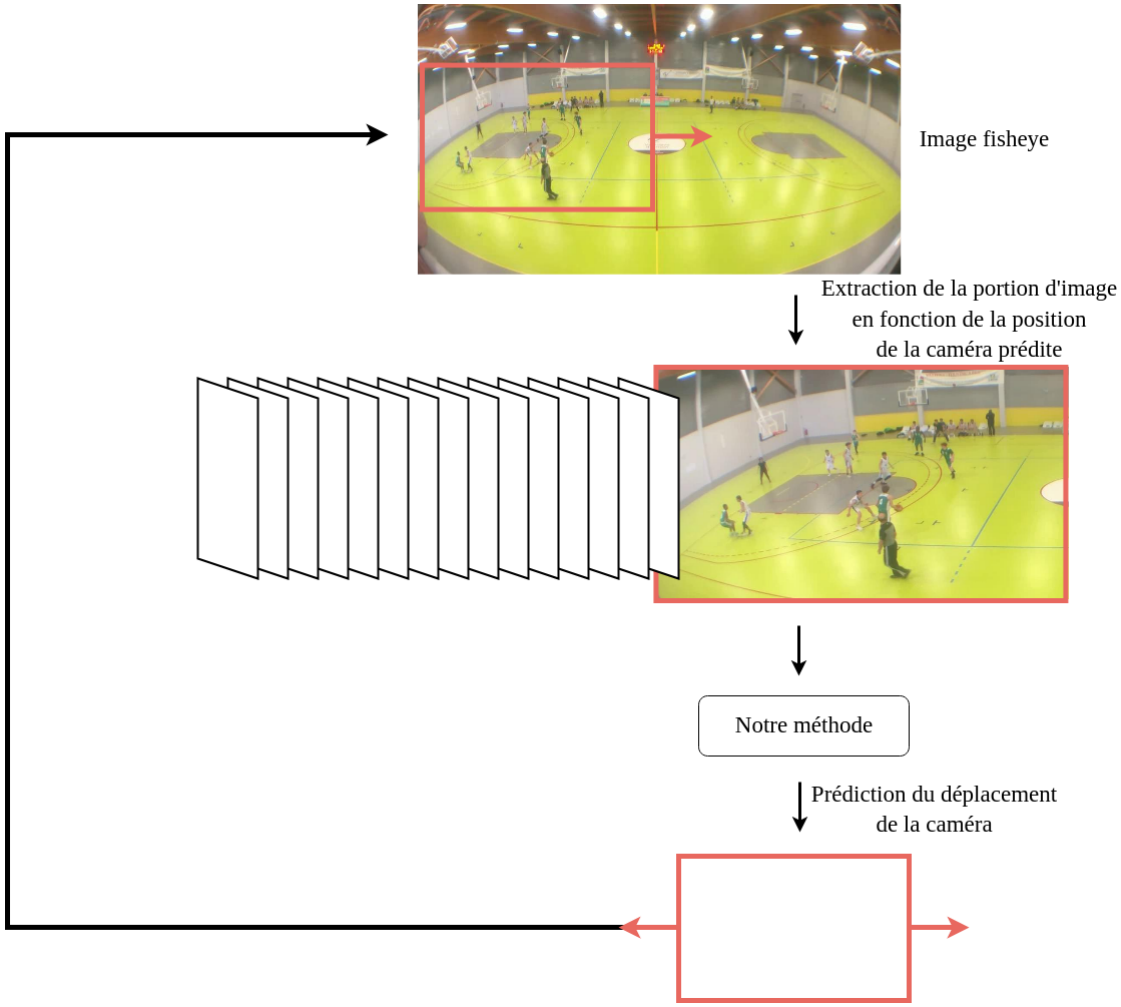


FIGURE 4.1 – **Schéma du pipeline de test sur les vidéos capturées avec une caméra fisheye.**

L'intérêt d'un tel modèle est d'augmenter le nombre de paramètres utilisés pour la prédiction et de s'affranchir de l'utilisation des situations qui posent problème lors des longues séquences. Ces travaux ont donné lieu à une pré-publication scientifique présentée dans Baldanza *et al.* (2022b).

4.2 Architecture CNN-3D pour le suivi d'action : NetCamMot

4.2.1 Reformulation

Nous reprenons la formulation du problème du chapitre précédent. Ici, on considère à nouveau les images comme des matrices. On définit, comme dans l'équation (3.2), $F^t \in \mathbb{R}^{N \times M \times 2}$ la matrice segmentée du flux optique calculé à l'instant t en fixant à 0

les éléments suffisamment proches de la médiane 2D (u^*, v^*) du flux optique. Ainsi, à chaque instant t, la matrice du flux optique du premier-plan est composée des éléments $f^t_{i,j}$ définis comme suit

$$f^t_{i,j} = \begin{cases} (u_{i,j} + u^*, v_{i,j} + v^*), & \text{si } (i,j) \in \Omega. \\ 0 & \text{sinon .} \end{cases} \tag{4.1}$$

où $\Omega = \{(i,j); ||(u_{i,j}, v_{i,j}) - (u^*, v^*)||^2 \geq \theta\}$, $(u_{i,j}, v_{i,j})$ étant le résultat de l'estimation du flux optique à la position i, j et θ le seuil définit comme dans l'équation (3.3). Cette matrice est ensuite normalisée avec le processus présenté dans la Section 3.3.2.

4.2.2 Détails de l'architecture

Ici, on remplace le modèle linéaire par morceaux présenté dans l'équation (3.6) par l'architecture **NetCamMot** (*Network for Camera Motion*). À chaque instant t le réseau prend en entrée n matrices du flux optique du premier plan F calculées avec des intervalles de k images. Le but est d'ajouter l'analyse de la dimension temporelle au réseau. L'écart k entre les entrées et le nombre n de flux optiques peut être ajusté en fonction des besoins de chaque sport. Dans notre cas, k et n ont été choisis expérimentalement comme les paramètres permettant d'obtenir les meilleures performances sur la base de données de test. Nous avons utilisé $k = 10$ et $n = 3$ pour les vidéos de basket-ball et de handball. Cela permet au modèle d'utiliser l'évolution des mouvements des joueurs dans le temps pour améliorer les performances. Ainsi, les prédictions dépendent également de ce qui s'est passé précédemment. Cela le rend également plus robuste aux variations d'environnement dûes au contexte amateur. Pour améliorer les temps de calcul, nous avons sous-échantillonné les images d'entrée à 160×90. Le réseau est entraîné sur les bases de données d'apprentissage présentées dans la Section 3.5 en utilisant l'optimiseur ADAM (Kingma et Ba, 2014).

L'architecture du réseau est résumée dans le Tableau 4.1. Elle est également schématisée dans la Figure 4.2.

L'architecture et le nombre de paramètres ont été choisis pour éviter le surapprentissage et pour être calculés en temps réel par des solutions embarquées. La prédiction est possible dès que le temps d'acquisition dépasse $(n-1)k$ images. Sur les CPU d'ordinateurs portables classiques, la sortie du réseau est calculée à 70,5 fps. Le temps de calcul dépend donc principalement de l'estimation du flux optique. Un des avantages de ce modèle est qu'il peut être évalué en temps réel pour des résolutions d'images classiques, mais aussi pour des solutions très haute résolution (type 4K) grâce au sous-échantillonnage. Ceci est également un avantage du fait de ne pas effectuer de détections d'objets com-

Block	Layers	output size
Conv1	Conv : 32 3x3 filters	(32, 90, 160)
	Maxpool : 2x2 filter	
Conv2	Conv : 64 3x3 filters	(64, 45, 80)
	Maxpool : 2x2 filter	
Conv3	Conv : 128 3x3 filters	(128, 22, 40)
	Maxpool : 2x2 filter	
Conv4	Conv : 256 3x3 filter	(256, 11, 20)
	Maxpool : 2x2 filter	
FC1	FC : 128x11x20	(1,128)
FC2	FC : 128x64	(1,64)
FC3	FC : 64x32	(1,32)
FC4	FC : 32x1	(1,1)

TABLEAU 4.1 – **Architecture du réseau.** Ce tableau détaille les différentes couches du réseau et les tailles des sorties pour des flux optiques segmentés de taille 90x160. L'entrée est traitée par 4 couches convolutives Conv1, Conv2, Conv3 et Conv4 produisant des cartes de caractéristiques convolutives. Le résultat est ensuite aplati pour passer par 4 couches entièrement connectées (FC1, FC2, FC3 et FC4).

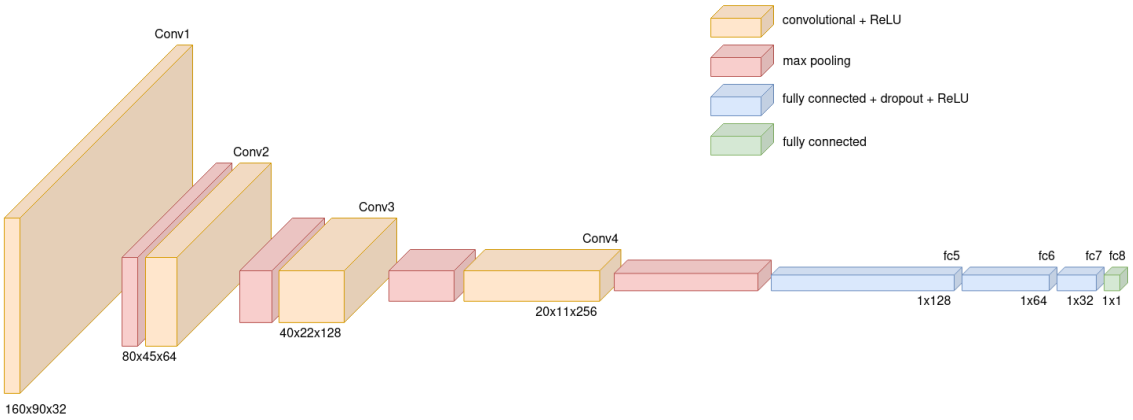


FIGURE 4.2 – **Schéma de l'architecture du réseau NetCamMot.**

plexes. La Figure 4.3 schématise le pipeline complet de la méthode pour comprendre son fonctionnement. Comme dans le chapitre précédent, la sortie du réseau est le mouvement prédit de la caméra, normalisé en pourcentage du terrain.

Cette méthode permet de prédire les mouvements verticaux et horizontaux. Cependant, comme dans le chapitre précédent, dans les sports étudiés ici, la composante verticale n'apporte pas de valeur ajoutée car toute la hauteur du terrain est capturée dans une seule image. C'est pourquoi nous ne prédirons que le mouvement horizontal dans la suite. Pour d'autres sports, le modèle peut facilement être adapté aux flux optiques bidimensionnels si la composante verticale doit être analysée.

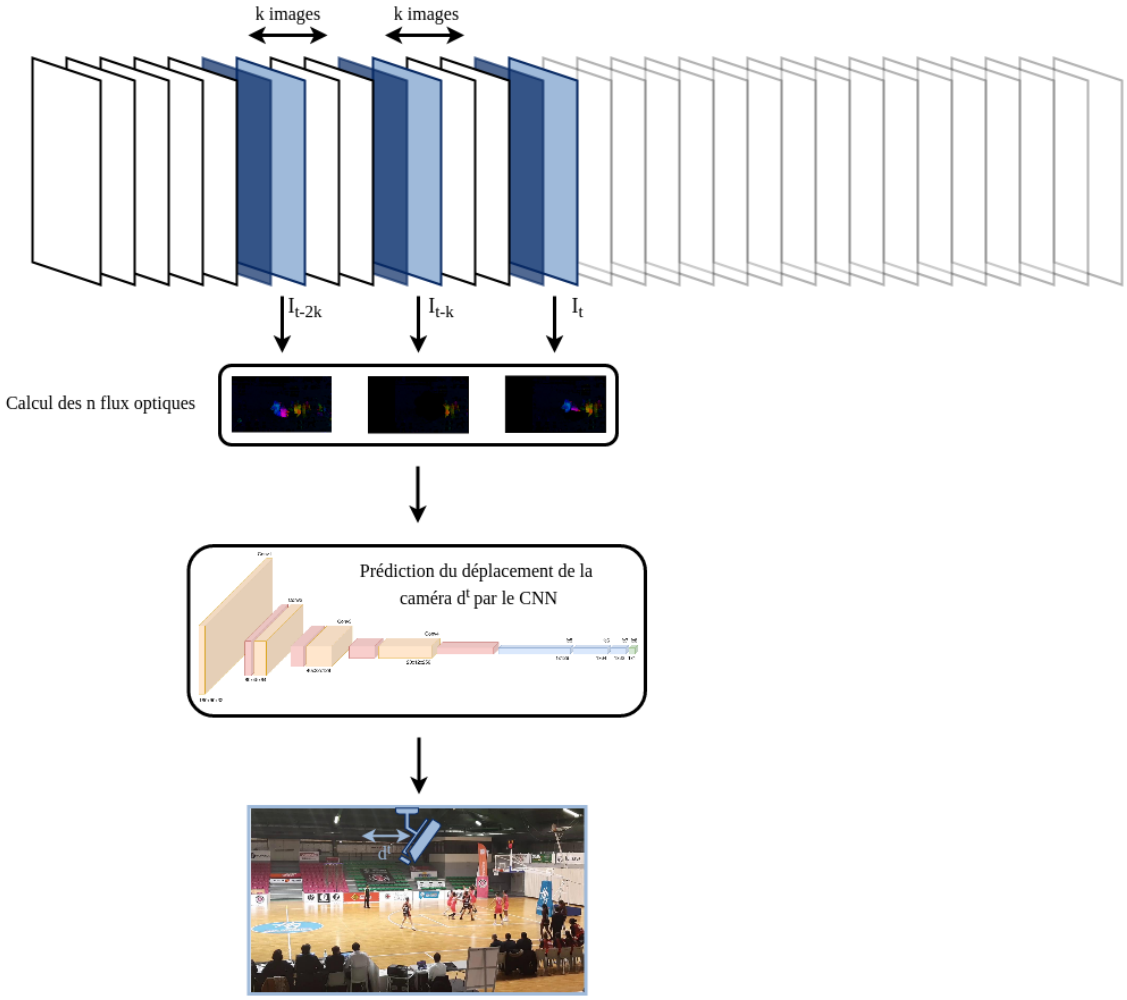


FIGURE 4.3 – **Pipeline détaillé de la méthode NetCamMot.**

4.3 Base de données de vidéos fisheye pour l'évaluation

Pour pouvoir évaluer notre méthode dans des conditions de suivi réelles, et pour que chaque prédiction ait une incidence sur la prochaine image analysée, nous disposons d'une base de données de vidéos capturées avec une caméra fisheye. Ce type de caméra est défini dans la Définition 4.1.1.

4.3.1 Vidéos

Le modèle étant entraîné sur des images classiques, pour que nous puissions nous servir de ces vidéos pour effectuer les tests, il faut que les images répondent à certains critères. Premièrement, il faut que la caméra soit placée suffisamment loin du terrain pour que les extrémités du terrain apparaissent et ne soient pas trop distordues par l'effet fisheye. De plus, il faut que la qualité et la résolution dans ces zones soit suffisamment bonne pour pouvoir faire nos calculs. Bien que notre algorithme soit capable de fonction-

ner pour des qualités variables, une résolution trop basse entraîne l'incapacité de fonctionner. La Figure 4.4 montre des exemples de portions d'images issues de vidéos que nous n'avons pas pu utiliser.

FIGURE 4.4 – **Exemples de portions d'images issues de vidéos inutilisables.** Ces images ne peuvent pas être analysées par notre méthode soit car la distorsion dans les angles déforme trop les joueurs (les deux images de gauche), soit car la qualité des portions est trop faible (les deux images de droite).

En respectant ces conditions, nous avons pu construire une base de données de test de plus de 10 minutes de match dans plusieurs gymnases.

4.3.2 Annotation

Chaque image de la base de données vidéo fisheye a été annotée manuellement via un algorithme conçu pour faciliter cette tâche où chaque annotation correspond aux coordonnées du centre de l'action dans la séquence. Pour diminuer la subjectivité de cette notion, l'annotation correspond aux coordonnées de la position du ballon sur le terrain lors des changements de coté et aux coordonnées du but/panier lorsqu'une équipe attaque le but ou le panier adverse. Expérimentalement, considérer ces points clés comme le centre de l'action permet d'obtenir une bonne capture du jeu, et sont habituellement choisis par l'humain en charge de capturer les actions. Cependant, le centre de l'action à filmer reste dépendant de l'interprétation humaine. Ce qui est universel dans notre cas, c'est la nécessité que ces points clés apparaissent à l'écran. Une telle annotation permet d'évaluer facilement s'ils sont présents dans l'image capturée par la caméra virtuelle.

4.4 Résultats

Comme expliqué dans l'introduction du chapitre, l'évaluation du modèle est découpée en deux parties. Dans un premier temps, nous allons évaluer le modèle sur la base de données présentée dans la Section 3.5. Évaluer le modèle sur cette base de données permet de comparer les résultats obtenus avec ceux du modèle linéaire par morceaux pour discuter des améliorations qu'il apporte. Dans un second temps, nous évaluerons le modèle sur les portions d'images fisheye pour évaluer la méthode en conditions réelles avec une caméra virtuelle.

4.4.1 Métrique d'évaluation

Pour évaluer le modèle sur la base de données de la Section 3.5, nous reprenons exactement les métriques du chapitre précédent. Nous utilisons la **MAE** et **l'erreur de suivi TE** définies dans la Section 3.6.1. Pour les vidéos longues de basket-ball, on évaluera également la **MAX TE**. Pour évaluer les résultats sur les vidéos fisheye, nous disposons de l'annotation manuelle du centre de l'action pour chaque image, présentée dans la section précédente. Ainsi, il nous est possible de déterminer avec précision s'il est visible dans l'image prédite par la méthode à chaque instant. Pour savoir si la caméra virtuelle perd le centre de l'action, il suffit de calculer la différence entre le centre de la portion d'image prélevée et l'annotation. Soit I_t une image fisheye de largeur N, $y^t \in [\![1; N]\!]$ l'annotation du centre de l'action dans I^t. On définit $c^t \in [\![1; N]\!]$ le centre de la portion d'image prélevée dans l'image fisheye. On cherche à évaluer

$$\delta^t = |y^t - c^t| \tag{4.2}$$

Soit $\tilde{N}$ la largeur de la portion d'image évaluée. Si la valeur de δ^t est supérieure à $\frac{\tilde{N}}{2}$, le centre de l'action n'apparaît pas à l'image. On considère alors cette image comme une erreur de suivi. Cette métrique est consistante car le centre de l'action annoté ne doit pas nécessairement être centré dans l'image pour que la vidéo soit considérée comme bien suivie. Dans les bases de données d'apprentissages notamment, il peut arriver que le centre de l'action tel que nous l'avons labellisé se rapproche de l'extrémité d'une image. Cependant, l'analyse des vidéos montre qu'il faut s'assurer que cet endroit du terrain apparaisse bien dans l'image pour que le suivi soit correct. L'évaluation de cette métrique sur nos images est schématisée dans la Figure 4.5. Grâce à cette métrique, on peut évaluer de façon précise la quantité d'action qui a été bien filmée par la caméra virtuelle. Dans cette partie on évaluera donc directement **la précision** du modèle, autrement dit le pourcentage de portions d'images qui contiennent le centre de l'action.

4.4.2 Comparaison avec le modèle linéaire par morceaux

Dans cette partie, nous comparons les résultats obtenus par NetCamMot avec le modèle linéaire par morceaux présenté dans le chapitre précédent. Pour justifier de l'utilisation de la dimension temporelle dans le modèle, nous comparons également NetCamMot à une version qui reprend l'architecture, mais n'utilise qu'une seule segmentation issue du moment t de la prédiction. Cette version du modèle s'appelle NetCamMotV0.

Les résultats de l'évaluation sur la base de données de test complète montrent que NetCamMot est plus précis que le modèle PEN (modèle linéaire par morceaux avec les fonctions de pénalisation dans la Loss d'apprentissage). Si on se concentre sur les per-

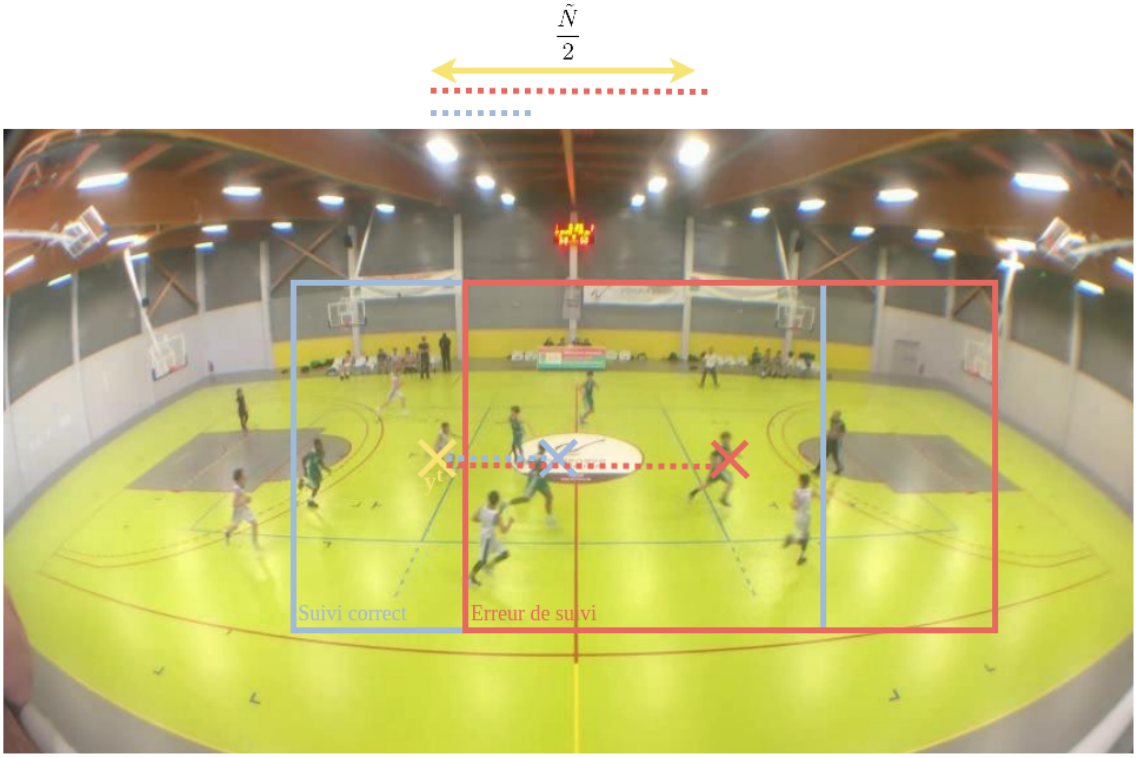


FIGURE 4.5 – **Évaluation de la précision du suivi dans les images fisheye.** Dans cet exemple, la portion d'image bleue est le résultat d'un suivi correct, le δ^t calculé à partir de son centre est inférieur à $\frac{\tilde{N}}{2}$. La portion d'image rouge est le résultat d'une erreur de suivi. La distance δ^t entre son centre et l'annotation y^t est supérieure à $\frac{\tilde{N}}{2}$.

formances moyennes sur chaque sous-base, elles sont toutes nettement meilleures avec le nouveau modèle. Également, plus de 75% des vidéos sont mieux traitées avec NetCamMot qu'avec PEN au sens des deux métriques. Ces résultats signifient que le nouveau modèle prédit des déplacements plus proches de la vérité terrain et qu'il reproduit par conséquent mieux le comportement d'acquisition humain que le modèle linéaire par morceaux.

Vidéos courtes de basket-ball

Les performances sur les vidéos courtes de basket-ball sont résumées dans le Tableau 4.2. Sur cette base de données, les moyennes des deux métriques évaluées sont nettement inférieures pour le nouveau modèle NetCamMot. Cela signifie que les mouvements de la caméra prédits sont en moyenne plus proches de la vérité terrain. La perte de l'action est alors moins probable avec cette méthode. Ces résultats montrent que l'utilisation d'un réseau neuronal convolutif avec un nombre accru de paramètres par rapport au modèle PEN améliore la précision et la robustesse du modèle. Notamment, l'augmentation du nombre de paramètres permet au modèle d'être plus flexible à l'apprentissage des données

	MAE			Tracking Error		
Video	PEN	NetCamMotV0	**NetCamMot**	PEN	NetCamMotV0	**NetCamMot**
1	**0.0001**	0.01	0.01	0.05	**0.0001**	0.001
2	0.16	**0.04**	0.06	**2.00**	4.41	2.28
3	0.3	0.08	**0.07**	9.12	**0.39**	2.90
4	0.39	0.11	**0.04**	24.72	2.51	**0.96**
5	0.25	0.14	**0.1**	36.62	31.15	**22.53**
6	0.27	**0.07**	0.15	37.92	**1.30**	4.91
7	0.19	**0.1**	0.11	13.01	3.28	**2.38**
8	0.12	**0.08**	0.09	2.78	**1.63**	2.06
9	**0.0001**	0.001	0.001	0.04	**0.001**	0.01
10	0.27	**0.08**	0.09	4.99	**1.64**	2.63
11	**0.0001**	0.001	0.01	0.14	**0.0001**	0.03
12	0.28	**0.11**	0.16	**6.66**	19.68	6.90
13	0.26	0.08	**0.05**	14.96	16.82	**0.69**
14	0.10	0.11	**0.04**	14.89	3.80	**1.46**
15	0.14	0.11	**0.06**	12.65	20.72	**11.13**
MEAN	0.18	0.075	**0.069**	12.04	7.16	**4.06**
BEST	3	**6**	**6**	2	**7**	6
NUMBER OF MAE >15				3	4	**1**

TABLEAU 4.2 – **Résultats des différents modèles sur la base de données des vidéos courtes de basket-ball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) obtenues par les modèles **NetCamMot**, NetCamMotV0 et PEN sur chaque vidéo de cette base de données. Le modèle NetCamMot parvient à reproduire l'acquisition réelle dans 14 vidéos sur 15.

variées.

En comparant NetCamMot avec la version NetCamMotV0, nous pouvons confirmer que l'ajout de la dimension temporelle en entrée augmente la précision du réseau sur cette base de données. Le modèle NetCamMot obtient une erreur moyenne sur cette base inférieure au sens de nos deux métriques. En étudiant les résultats vidéo par vidéo, le modèle NetCamMotV0 obtient de meilleurs résultats au sens de la MAE classique. Cependant, la moyenne de cette erreur sur toute la base de données est plus élevée. On peut déduire de cette observation que sur certaines vidéos, le modèle NetCamMotV0 obtient des résultats nettement moins bons. Cette analyse permet de confirmer que l'ajout de la dimension temporelle augmente la robustesse du modèle face aux variations d'environnements entre les vidéos. L'étude de l'erreur de suivi appuie ce constat. NetCamMotV0 obtient une erreur de suivi supérieure à 15% du terrain sur 4 vidéos contre une seule pour le modèle NetCamMot.

Vidéos courtes de handball

	MAE			Tracking Error		
Video	PEN	NetCamMotV0	**NetCamMot**	PEN	NetCamMotV0	**NetCamMot**
1	0.21	0.11	**0.08**	**4.92**	10.03	5.68
2	0.31	0.13	**0.11**	**10.68**	17.86	10.79
3	0.10	**0.06**	0.11	14.04	**3.14**	5.31
4	0.1	**0.07**	0.16	4.02	3.68	**2.99**
5	0.19	**0.05**	0.29	9.27	**3.97**	9.09
6	0.15	0.13	**0.1**	**4.82**	21.53	11.6
7	0.1	**0.07**	0.08	5.14	12.66	**2.61**
8	0.26	**0.08**	0.17	9.36	7.96	**5.49**
9	0.14	**0.1**	**0.1**	**2.97**	20.45	6.27
10	0.15	**0.12**	0.13	**6.17**	22.35	11.22
11	0.24	**0.06**	0.11	9.65	13.3	**6.62**
12	0.22	**0.09**	0.1	7.02	10.43	**2.71**
13	0.14	**0.05**	0.12	4.1	8.25	**2.8**
14	0.15	0.08	**0.07**	4.80	13.13	**0.57**
MEAN	0.18	**0.08**	0.12	6.93	12.05	**5.98**
BEST	0	**9**	5	5	2	**7**
NUMBER OF MAE >15				**0**	4	**0**

TABLEAU 4.3 – **Résultats des différents modèles sur la base de données des vidéos courtes de handball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) obtenues par les modèles **NetCamMot**, NetCamMotV0 et PEN sur chaque vidéo de cette base de données. Le modèle NetCamMot parvient à reproduire l'acquisition réelle dans 15 vidéos sur 15.

Les résultats sur les vidéos courtes de handball présentés dans le Tableau 4.3 montrent également que le modèle NetCamMot est la méthode la plus précise au sens de l'erreur de suivi. L'étude de cette métrique montre qu'il arrive à reproduire l'acquisition de la vérité terrain dans toutes les vidéos. Bien que NetCamMotV0 obtienne des prédictions locales plus précises, l'étude de l'erreur de suivi permet de constater qu'elle est beaucoup plus sujette au phénomène de dérive et obtient des résultats moins bons dans le temps. Ici aussi, ce modèle obtient une erreur de suivi supérieure à 15% dans 4 vidéos sur 14, contre 0 pour NetCamMot. Une fois encore, ces résultats soulignent l'intérêt d'utiliser la composante temporelle pour la prédiction, et particulièrement son impact sur la robustesse de la méthode aux variations entre les environnements.

Sur cette base de données aussi, NetCamMot obtient de meilleurs résultats que le modèle PEN au sens des deux métriques.

		MAE		Tracking Error		MAX TE	
Video	Duration	PEN	**NetCamMot**	PEN	**NetCamMot**	PEN	**NetCamMot**
1	0 :34	0.15	**0.09**	6.67	**2.42**	10.69	**3.76**
2	0 :24	**0.001**	0.003	0.05	**0.005**	0.057	**0.005**
3	2 :18	0.1	**0.07**	14.14	**0.03**	45.86	**12.58**
4	0 :30	**0.001**	0.004	0.11	**0.01**	0.12	**0.009**
5	1 :42	0.12	**0.04**	23.08	**5.68**	39.50	**8.87**
6	0 :53	0.16	**0.07**	11.21	**4.44**	17.72	**10.83**
7	0 :56	0.14	**0.08**	24.15	**4.64**	53.82	**9.24**
	MEAN	0.1	**0.05**	11.34	**2.46**	-	-
	BEST	2	**5**	0	**7**	0	**7**

TABLEAU 4.4 – **Résultats des modèles NetCamMot et PEN sur la base de données des vidéos longues de basket-ball.** Erreur absolue moyenne calculée sur les prédictions simples (MAE) et sur les prédictions intégrées en fonction du temps (Tracking Error) obtenues par les modèles **NetCamMot** et PEN sur chaque vidéo de cette base de données. Le modèle NetCamMot parvient à reproduire l'acquisition réelle dans 7 vidéos sur 7.

Vidéos longues de basket-ball

Les performances de la méthode sur les longues séquences de basket-ball sont présentées dans le Tableau 4.4. Ici, nous ne comparons plus que NetCamMot avec PEN. Ces résultats ainsi que les courbes présentées dans la Figure 4.6 montrent que NetCamMot obtient des résultats précis tout au long des longues séquences de matchs. Pour être cohérents avec la métrique d'évaluation définie dans le chapitre précédent sur ces vidéos, nous évaluons l'erreur de suivi maximum sur les sous-séquences de 10 secondes. On constate que le modèle NetCamMot n'obtient aucune erreur de suivi supérieure à 15% sur les sous-séquences de la base. Ces résultats mettent également en évidence une amélioration apportée par le réseau de neurones convolutif par rapport à la méthode linéaire par morceaux PEN. En effet, l'utilisation des réseaux permet de corriger une des grandes limites de ce modèle qui ne parvenait pas à obtenir de bons résultats sur les longues séquences.

Les résultats présentés dans la Figure 4.6 montrent comment le modèle NetCamMot (courbe bleue) est capable de prédire des valeurs proches de la vérité terrain (courbe noire) par rapport aux autres modèles. Le nombre d'images est différent entre les séquences courtes et les séquences longues, ce qui explique pourquoi les transitions semblent plus courtes sur les courbes associées aux séquences longues. Sur les images, les rectangles bleus permettent de voir le décalage entre les prédictions et la vérité terrain. Ces exemples permettent de voir à quel point il est faible.

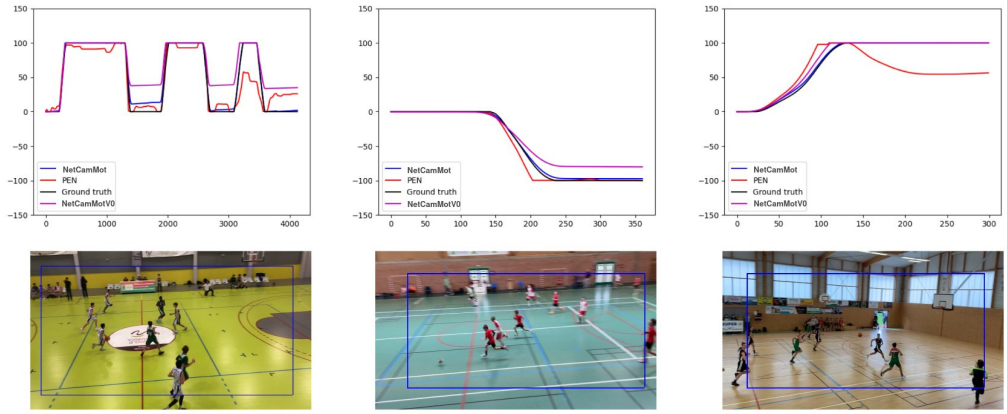


FIGURE 4.6 – **Exemples de vidéos bien suivies par NetCamMot.** À gauche, une longue séquence de basket-ball, au milieu, une séquence de handball et à droite, une courte séquence de basket-ball. Le rectangle bleu montre le décalage de la position de la caméra prédite par rapport à l'acquisition réelle.

4.4.3 Caméra virtuelle

Processus de correction

Pour reproduire des conditions réelles et améliorer nos performances, nous avons ajouté un processus de récupération de l'action en cas d'erreur. On considère ici que si le flux optique segmenté F^t définit à l'équation (4.1) ne contient que des 0, c'est qu'aucun joueur ne figure à l'écran et que le fil de l'action a été perdu par la caméra virtuelle. Pour corriger le suivi lorsque ce phénomène se produit, nous replaçons la caméra directement à l'extrémité opposée du terrain. En considérant la caméra comme contenant la moitié du terrain, si aucun joueur n'apparaît à l'image, c'est qu'ils sont dans la moitié de terrain opposée. Un exemple de ce cas est présenté dans la Figure 4.7, il est repérable grâce à la mention "Correction".

Résultats de suivi

Les résultats du modèle sur les portions d'images fisheye de la Section 4.3 sont présentés dans le Tableau 4.5. Les tests sont répartis sur 7 vidéos. Dans ce tableau, les lettres BB et HB indiquent s'il s'agit respectivement d'une vidéo de basket-ball ou de handball. Ces résultats montrent que le modèle est capable de calculer un suivi précis de l'action lorsque les images capturées dépendent des prédictions précédentes. Le modèle parvient à obtenir de bons résultats dans plus de 89% des images. L'étude des vidéos capturées par la caméra virtuelle générées par notre méthode permet de valider l'interprétation des performances via notre métrique. En effet, les vidéos montrent que le modèle parvient très bien à reproduire l'acquisition humaine du sport amateur.

Sport	Durée	Nb d'images	Précision (%)
BB	1 :30	2700	92.18
BB	0 :47	1400	87.86
BB	1 :57	3500	89.83
BB	0 :50	1500	98.73
BB	1 :27	2600	79.42
HB	1 :14	2200	96.14
HB	2 :57	5300	88.09
Total	10 :42	19200	**89.55**

TABLEAU 4.5 – **Résultats du modèle NetCamMot sur la base de données des vidéos fisheye.** Les lettres BB désignent les vidéos de basket-ball et les lettres HB les vidéos de handball. Une image est considérée comme un bon résultat de suivi si le centre de l'action annoté figure dans la portion d'image suivie par le modèle. NetCamMot obtient plus de 89% de séquences bien suivies.

Les prédictions étant calculées sur des portions d'images, la résolution des images étudiées est globalement faible. Obtenir de tels résultats sur ce type d'images confirme que notre méthode est robuste dans ces conditions. De plus, les vidéos de cette base de données sont captées dans des gymnases différents à des distances variables du bord du terrain. Or, la méthode obtient de bons résultats de suivi dans toutes les vidéos. Ces performances justifient également de la robustesse de la méthode aux variations amenées par les conditions amateur.

Comportements et limites du modèle

On constate en observant les courbes oranges des Figures 4.7 et 4.8, que lors des changements de côté, les déplacements de la caméra prédits sont quasiment linéaires jusqu'à atteindre l'autre extrémité du terrain. Une telle observation peut s'expliquer de trois façons. Premièrement, il est préférable pour l'œil que la caméra suive un mouvement linéaire, même si le centre de l'action marque un léger arrêt. C'est généralement avec ce type de mouvements lissés que sont capturées les vidéos de sport, notamment dans notre base de données d'apprentissage. Cela explique que le modèle ait appris ce comportement.

Ensuite, il est possible que la base de données d'apprentissage soit légèrement biaisée du fait que les vidéos de la plateforme Rematch sont courtes. Les vidéos étant courtes, il est plus compliqué de trouver des vidéos avec des arrêts de caméra au milieu du terrain qui en parcourent l'intégralité. Or, nous avions besoin que la caméra parcoure le terrain en entier afin de pouvoir normaliser les déplacements en pourcentage du terrain.

Enfin, il peut arriver que les mouvements des joueurs ne soient pas parfaitement décor-

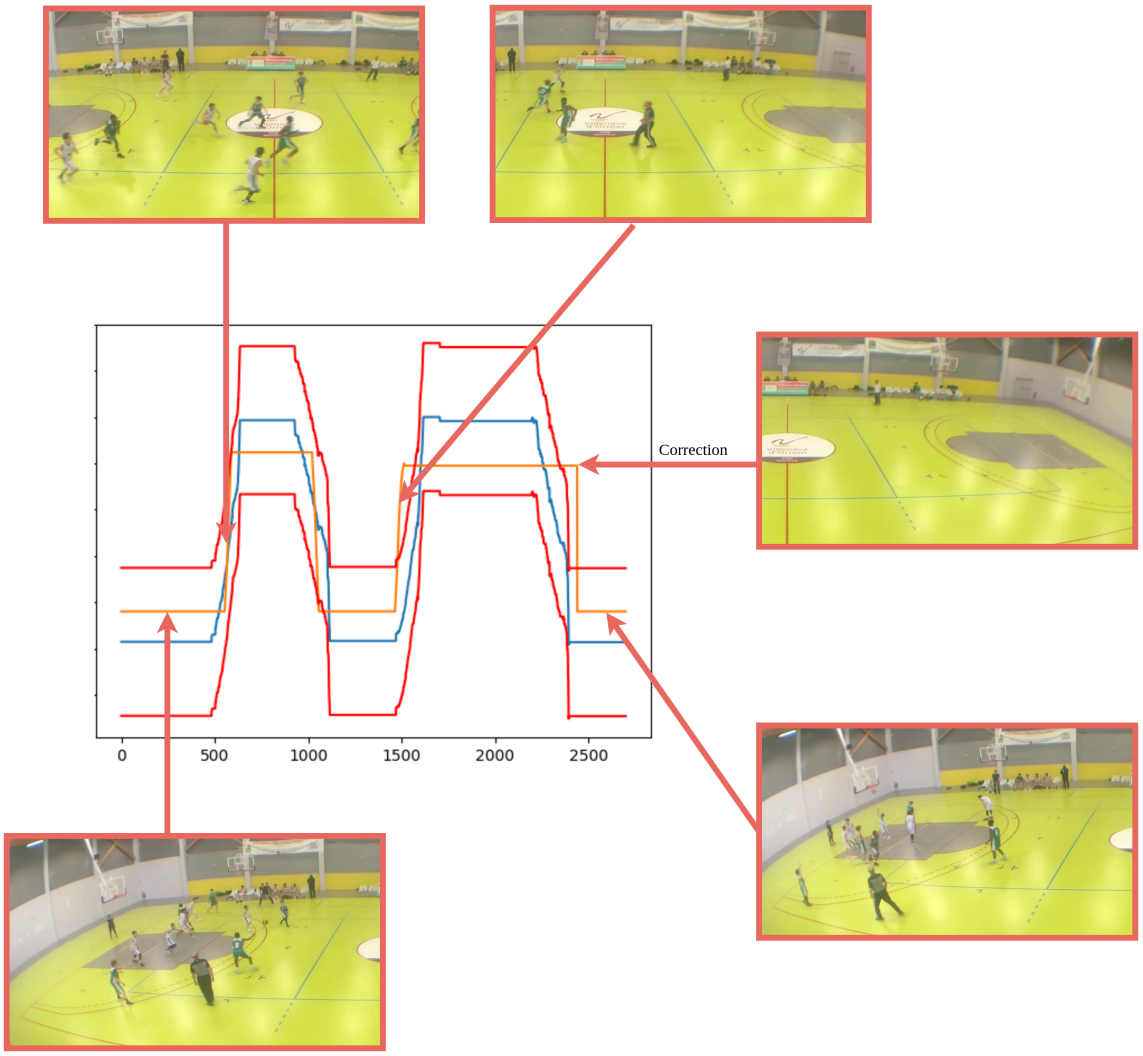


FIGURE 4.7 – **Courbes des résultats obtenus par NetCamMot sur une vidéo de basket-ball.** Sur ce graphique, la courbe bleue correspond à l'annotation manuelle. La courbe orange correspond à la position de la caméra prédite. Les deux courbes rouges correspondent aux limites à ne pas franchir pour que le centre de l'action annoté soit dans le cadre. Les images autour correspondent aux portions d'images fisheye extraites au moment clé indiqué par la flèche.

rélés des mouvements de la caméra et que les mouvements précédemment prédits aient un léger impact sur le flux optique segmenté de l'image étudiée. Ceci peut expliquer que les mouvements de la caméra soient linéaires alors que les déplacements de l'action annotés ne le sont pas.

Ce comportement peut avoir une incidence sur le suivi dans le cas où le jeu changerait de sens au cours d'un déplacement. Heureusement, dans ce type de sport, c'est un événement assez rare qui peut rapidement être compensé par le processus de correction d'erreur.

Il peut également arriver que le modèle rencontre des limites, notamment lorsqu'il

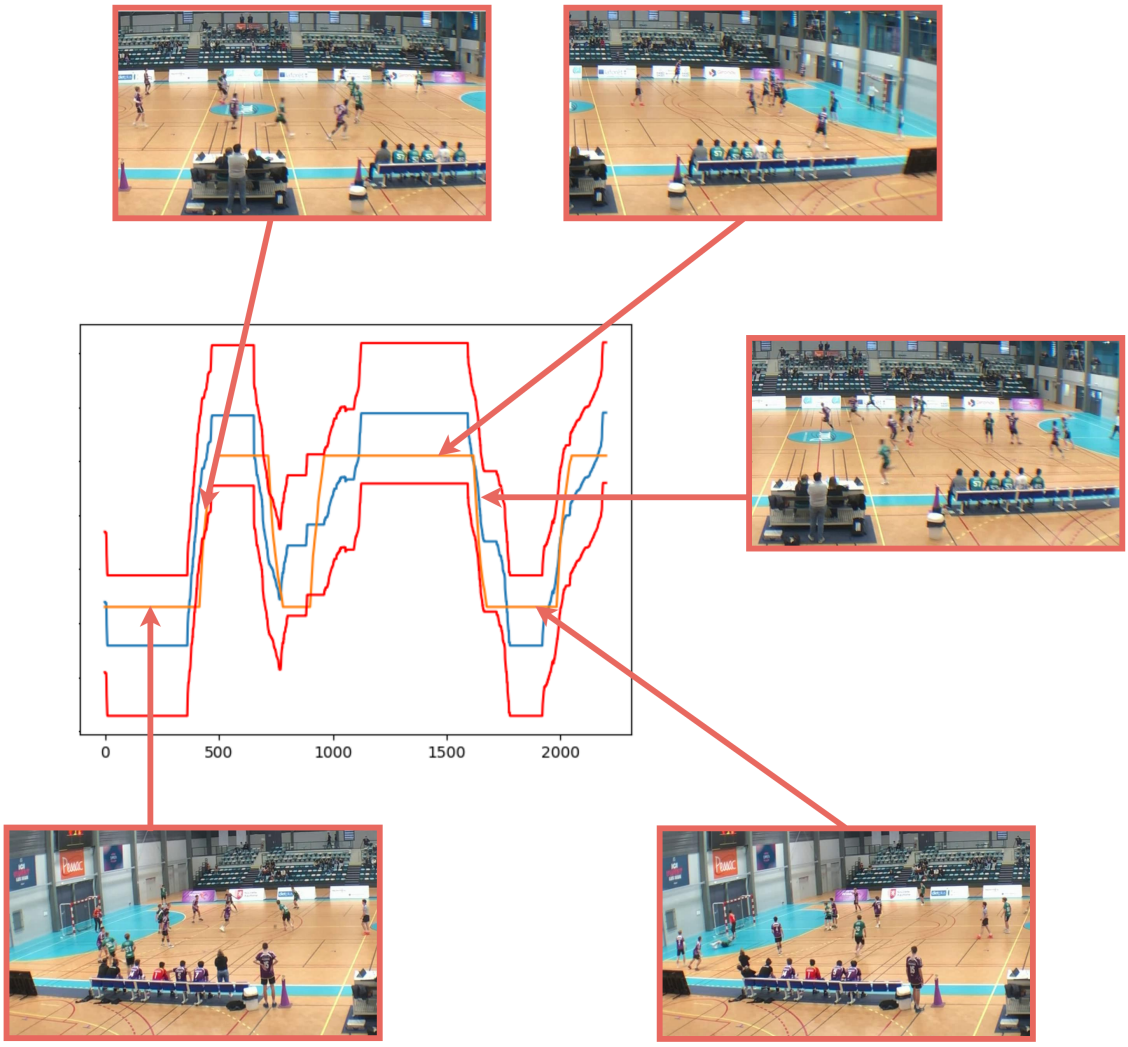


FIGURE 4.8 – **Courbes des résultats obtenus par NetCamMot sur une vidéo de handball.** Sur ce graphique, la courbe bleue correspond à l'annotation manuelle. La courbe orange correspond à la position de la caméra prédite. Les deux courbes rouges correspondent aux limites à ne pas franchir pour que le centre de l'action annoté soit dans le cadre. Les images autour correspondent aux portions d'images fisheye extraites au moment clé indiqué par la flèche.

reste fixe par erreur dans une zone proche du milieu du terrain qui lui permet de distinguer certains joueurs, mais pas le centre de l'action. Il ne parvient alors pas à détecter son erreur et à se replacer. Il arrive également que le modèle perde le centre de l'action pendant quelques dixièmes de secondes lors des changements de côté, comme on peut le voir sur les Figures 4.7 et 4.8. Observer ce comportement est logique. Le ballon n'étant pas détecté et les joueurs s'étalant sur une surface de terrain plus grande que celle présente à l'écran, il est normal que l'étude seule des déplacements de ceux qui figurent à l'écran ne suffise pas à obtenir un résultat plus précis. Ce phénomène n'entraîne cependant pas la perte complète du fil de l'action et on constate que le ballon réapparaît rapidement à

l'écran dans ce type de cas.

4.5 Conclusion du chapitre

Dans ce chapitre, nous présentons une nouvelle méthode de suivi d'action dans les vidéos de sport amateur visant à être embarquée dans une solution de caméra PTZ. Cette méthode est basée sur l'utilisation d'un réseau neuronal convolutif 3D pour permettre d'analyser la composante temporelle. Cette méthode a montré des résultats très performants sur toutes nos bases de données de tests. Même lorsque ses prédictions ont un impact sur les prochaines images capturées, il parvient à suivre le jeu dans plus de 89% des cas. De plus, baser les prédictions sur les mouvements des joueurs dans l'image permet de définir un processus de correction en cas d'erreur. Cette méthode a montré cependant certaines limites. Notamment, il peut arriver que la caméra prédite perde le ballon quelques fractions de seconde lors de ses déplacements. De plus, le processus de correction peut montrer des failles lorsqu'une partie des joueurs apparaît toujours à l'écran. Il peut être intéressant de concevoir une solution embarquée qui calcule le suivi sur une image capturée avec un objectif muni d'une distance focale plus courte, ou de plusieurs caméras. Ainsi, les calculs peuvent être réalisés sur une région plus grande, voir sur l'intégralité du terrain pour pouvoir être plus robustes face à ce type d'erreurs.

Chapitre 5

Conclusion

Sommaire

5.1 Récapitulatif des contributions 98

5.1.1 Proposition temporelle d'actions 98

5.1.2 Suivi d'action vidéo 98

5.1.3 Publications 98

5.1.4 Intégrations 98

5.2 Perspectives et ouvertures 99

Dans ce manuscrit, nous avons abordé les problématiques de proposition temporelle d'actions et de suivi d'action vidéo dans le sport amateur. Ces travaux ont pour but de permettre la découpe automatique des vidéos de la plateforme Rematch en fonction de leur contenu ainsi que la conception d'une caméra autonome visant à l'acquisition automatique du sport amateur.

Le Chapitre 2 traite la problématique de proposition temporelle d'actions pour le découpage automatique des vidéos. Il présente une méthode qui permet la localisation précise du contenu d'intérêt en limitant le nombre de faux négatifs.

Les Chapitres 3 et 4 traitent la problématique de suivi d'action vidéo. Le Chapitre 3 présente les bases d'une nouvelle approche permettant de répondre à ce problème. Il s'agit d'une méthode à un étage qui prédit directement les déplacements de la caméra nécessaires à suivre l'action à partir des mouvements des joueurs dans l'image. Cette méthode utilise un modèle d'apprentissage supervisé linéaire par morceaux pour prédire les déplacements. Le Chapitre 4 reprend l'approche définie dans le chapitre précédent, mais utilise un réseau de neurones convolutif 3D, NetCamMot, pour calculer le mouvement de la caméra. Ce chapitre présente également les résultats obtenus sur des portions d'images capturées avec une caméra fisheye, dans lesquelles il est possible de déplacer une caméra virtuelle à l'intérieur du terrain comme avec une caméra PTZ.

5.1 Récapitulatif des contributions

5.1.1 Proposition temporelle d'actions

Concernant la problématique de proposition temporelle d'actions, nous avons présenté une méthode qui permet de fournir des bornes de localisation précises de l'action tout en permettant de limiter le nombre de faux négatifs dans les résultats. Ceci est rendu possible grâce à la définition d'un processus de proposition/validation des bornes. La limitation du nombre de faux négatifs a pour objectif de limiter le nombre de vidéos endommagées lorsqu'on se sert de la proposition temporelle d'actions pour découper une vidéo en fonction de son contenu. L'avantage de ce processus est qu'il peut être adapté pour fonctionner avec d'autres méthodes à condition de redéfinir des descripteurs adaptés à la tâche visée.

5.1.2 Suivi d'action vidéo

Dans cette thèse, nous proposons une nouvelle approche pour le suivi d'action vidéo visant à être embarquée dans des solutions d'acquisition autonomes. Les méthodes de la littérature, notamment dans le contexte sportif, basent la direction des caméras et le suivi vidéo sur le suivi des objets dans les séquences. Pour répondre aux problématiques amenées par le contexte du sport amateur qui rend inutilisable ce type de méthode, nous avons défini une méthode qui n'utilise aucune détection d'objet. Notre méthode calcule les déplacements des joueurs dans l'image grâce à une segmentation du flux optique et prédit les déplacements de la caméra en fonction de ces derniers. Cette méthode présente une approche nouvelle qui répond au contexte du sport amateur et permet également l'annotation automatique des bases de données d'apprentissage.

5.1.3 Publications

Ces travaux ont donné lieu à deux publications scientifiques et une pré-publication. Concernant les articles publiés, il s'agit d'une publication dans une conférence nationale (ORASIS) et d'une publication dans une conférence internationale (EUSIPCO).

5.1.4 Intégrations

Les méthodes présentées dans cette thèse ont également été intégrées dans la conception de produits industriels. La méthode permettant le découpage automatique des vidéos est actuellement intégrée dans le pipeline d'encodage vidéo de l'application Rematch. La méthode de suivi vidéo n'a pas encore été intégrée dans un produit fini, mais joue un

rôle prédominant dans la conception d'une solution d'acquisition autonome en développement.

5.2 Perspectives et ouvertures

Les travaux concernant le suivi d'action ont montré certaines limites dans leur capacité à suivre le centre de l'action en continu. Pour améliorer les performances de la méthode, il est possible d'utiliser de nouveaux opérateurs permettant d'affiner notre approche. Il serait intéressant de voir comment réagit un réseau de neurones récurrent (RNN) pour notre tâche. L'utilisation de ce type de réseaux permettrait de maintenir un état interne et de prendre en compte les informations précédentes pour calculer les nouvelles prédictions. Ces réseaux sont fréquemment utilisés dans des solutions embarquées d'analyse vidéo, notamment car ils adaptent les nouvelles prédictions aux prédictions précédentes. Dans notre cas, cela pourrait être utile car il permettrait d'adapter la vitesse des déplacements de la caméra en fonction des déplacements précédents. De plus, le maintien d'un état interne permettrait dans notre cas d'intégrer la notion de position de la caméra afin de mieux adapter les prédictions à chaque situation. Également, dans ce type de solutions, les filtres de Kalman sont également souvent utilisés. Ils présentent également un axe d'amélioration envisageable pour notre solution car ils permettent de maintenir une estimation de l'état du système pour limiter les erreurs de prédictions qui peuvent avoir des conséquences très fortes dans ce type d'applications. D'autre part, l'utilisation des caméras PTZ pour la capture du sport amateur engendre l'incapacité de calculer la localisation à partir d'une information omnisciente. Il est impossible de contenir tous les joueurs dans l'image à chaque instant. Avec ce type de caméra, lors d'une erreur de suivi, les calculs devant être réalisés à partir de l'image capturée, il peut être compliqué de retrouver rapidement la bonne zone du terrain. Il peut être intéressant d'équiper la solution avec une caméra qui permet d'analyser des images contenant le terrain dans son intégralité. Dans ce sens, il serait judicieux d'équiper la solution avec une caméra fisheye ou avec plusieurs caméras pour l'acquisition de l'image qui permet le calcul du centre de l'action. Cela permettrait d'observer les résultats d'un modèle qui base le suivi de l'action sur les positions et les déplacements des joueurs sur le terrain complet, en disposant d'une information sur chaque joueur. Si l'étude des déplacements des joueurs dans une image classique parvient à donner des résultats aussi élevés, ajouter à l'analyse une information de position dans le référentiel du terrain ainsi qu'une connaissance complète sur les déplacements des joueurs devrait obtenir des résultats très précis. C'est d'ailleurs sur ce type d'étude que devrait se concentrer la suite des travaux de vision par ordinateur de l'entreprise Rematch.

Bibliographie

J. G. ALLEN, R. Y. XU, J. S. JIN *et al.* : Object tracking using camshift algorithm and multiple quantized feature spaces. *In ACM international conference proceeding series*, vol. 100, p. 3–7. Citeseer, 2004.

A. D. BAGDANOV, A. DEL BIMBO et W. NUNZIATI : Improving evidential quality of surveillance imagery through active face tracking. *In 18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, p. 1200–1203. IEEE, 2006.

A. BALDANZA, J.-F. AUJOL, Y. TRAONMILIN et F. ALARY : Découpage automatique de vidéos de sport amateur par détection de personnes et analyse de contenu colorimétrique. *In ORASIS 2021*, 2021.

A. BALDANZA, J.-F. AUJOL, Y. TRAONMILIN et F. ALARY : Piecewise linear prediction model for action tracking in sports. *Proceedings of the 30th European Signal Processing Conference (EUSIPCO 2022)*, 2022a.

A. BALDANZA, J. F. AUJOL, Y. TRAONMILIN et F. ALARY : Real-time multi-sport action tracking with convolutional neural networks. 2022b.

S. BELL, C. L. ZITNICK, K. BALA et R. GIRSHICK : Inside-outside net : Detecting objects in context with skip pooling and recurrent neural networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2874–2883, 2016.

P. BERGMANN, T. MEINHARDT et L. LEAL-TAIXE : Tracking without bells and whistles. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, p. 941–951, 2019.

L. BERTINETTO, J. VALMADRE, J. F. HENRIQUES, A. VEDALDI et P. H. TORR : Fully-convolutional siamese networks for object tracking. *In Computer Vision–ECCV 2016 Workshops : Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, p. 850–865. Springer, 2016.

A. BEWLEY, Z. GE, L. OTT, F. RAMOS et B. UPCROFT : Simple online and realtime tracking. *In 2016 IEEE international conference on image processing (ICIP)*, p. 3464–3468. IEEE, 2016.

J. BROMLEY, I. GUYON, Y. LECUN, E. SÄCKINGER et R. SHAH : Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993.

M. BURIC, M. IVASIC-KOS et M. POBAR : Player tracking in sports videos. *In 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, p. 334–340. IEEE, 2019.

M. BURIĆ, M. POBAR et M. IVAŠIĆ-KOS : Object detection in sports videos. *In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, p. 1034–1039. IEEE, 2018.

J. CARREIRA et A. ZISSERMAN : Quo vadis, action recognition ? a new model and the kinetics dataset. *In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 6299–6308, 2017.

Y.-W. CHAO, S. VIJAYANARASIMHAN, B. SEYBOLD, D. A. ROSS, J. DENG et R. SUKTHANKAR : Rethinking the faster r-cnn architecture for temporal action localization. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 1130–1139, 2018.

C. CORTES et V. VAPNIK : Support-vector networks. *Machine learning*, 20:273–297, 1995.

T. COVER et P. HART : Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

N. DALAL et B. TRIGGS : Histograms of oriented gradients for human detection. *In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, p. 886–893. Ieee, 2005.

J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI et L. FEI-FEI : Imagenet : A large-scale hierarchical image database. *In 2009 IEEE conference on computer vision and pattern recognition*, p. 248–255. Ieee, 2009.

T. D'ORAZIO, C. GUARAGNELLA, M. LEO et A. DISTANTE : A new algorithm for ball recognition using circle hough transform and neural classifier. *Pattern recognition*, 37 (3):393–408, 2004.

A. DOSOVITSKIY, L. BEYER, A. KOLESNIKOV, D. WEISSENBORN, X. ZHAI, T. UNTERTHINER, M. DEHGHANI, M. MINDERER, G. HEIGOLD, S. GELLY *et al.* : An image is worth 16x16 words : Transformers for image recognition at scale. *arXiv preprint arXiv :2010.11929*, 2020.

E. DUBROFSKY : Homography estimation. *Diplomová práce. Vancouver : Univerzita Britské Kolumbie*, 5, 2009.

M. EVERINGHAM, L. VAN GOOL, C. K. WILLIAMS, J. WINN et A. ZISSERMAN : The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009.

M. EVERINGHAM, L. VAN GOOL, C. K. WILLIAMS, J. WINN et A. ZISSERMAN : The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

B. G. FABIAN CABA HEILBRON, Victor Escorcia et J. C. NIEBLES : Activitynet : A large-scale video benchmark for human activity understanding. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 961–970, 2015.

A. GEIGER, P. LENZ, C. STILLER et R. URTASUN : Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

R. GIRSHICK : Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision*, p. 1440–1448, 2015.

R. GIRSHICK, J. DONAHUE, T. DARRELL et J. MALIK : Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 580–587, 2014.

M. A. HAQUE, K. NASROLLAHI et T. B. MOESLUND : Real-time acquisition of high quality face sequences from an active pan-tilt-zoom camera. *In 2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, p. 443–448. IEEE, 2013.

K. HE, X. ZHANG, S. REN et J. SUN : Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770–778, 2016.

W. HE, T. YAMASHITA, H. LU et S. LAO : Surf tracking. *In 2009 IEEE 12th International Conference on Computer Vision*, p. 1586–1592. IEEE, 2009.

D. HELD, S. THRUN et S. SAVARESE : Learning to track at 100 fps with deep regression networks. *In Computer Vision–ECCV 2016 : 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, p. 749–765. Springer, 2016.

P. W. HOLLAND et R. E. WELSCH : Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

A. G. HOWARD, M. ZHU, B. CHEN, D. KALENICHENKO, W. WANG, T. WEYAND, M. ANDREETTO et H. ADAM : Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv :1704.04861*, 2017.

M.-C. HU, M.-H. CHANG, J.-L. WU et L. CHI : Robust camera calibration and player tracking in broadcast basketball video. *IEEE Transactions on Multimedia*, 13(2):266–279, 2010.

G. HUANG, Z. LIU, L. VAN DER MAATEN et K. Q. WEINBERGER : Densely connected convolutional networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4700–4708, 2017.

S. HURAULT, C. BALLESTER et G. HARO : Self-supervised small soccer player detection and tracking. *In Proceedings of the 3rd international workshop on multimedia content analysis in sports*, p. 9–18, 2020.

S. W. IBRAHIM : A comprehensive review on intelligent surveillance systems. *Communications in science and technology*, 1(1), 2016.

P. JACCARD : Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

J. JANAI, F. GÜNEY, A. BEHL, A. GEIGER *et al.* : Computer vision for autonomous vehicles : Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

Y.-G. JIANG, J. LIU, A. ROSHAN ZAMIR, G. TODERICI, I. LAPTEV, M. SHAH et R. SUKTHANKAR : THUMOS challenge : Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/, 2014.

G. JOCHER, A. CHAURASIA, A. STOKEN, J. BOROVEC, NANOCODE012, Y. KWON, K. MICHAEL, TAOXIE, J. FANG, IMYHXY, LORNA, Z. YIFU, C. WONG, A. V, D. MONTES, Z. WANG, C. FATI, J. NADAR, LAUGHING, UNGLVKITDE, V. SONCK, TKIANAI, YXNONG, P. SKALSKI, A. HOGAN, D. NAIR, M. STROBEL et M. JAIN : ultralytics/yolov5 : v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, 2022.

R. E. KALMAN : A new approach to linear filtering and prediction problems. 1960.

P. R. KAMBLE, A. G. KESKAR et K. M. BHURCHANDI : A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*, 27(1):58–69, 2019.

S. J. KIM, J.-Y. NAM et B. C. KO : Online tracker optimization for multi-pedestrian tracking using a moving vehicle camera. *IEEE Access*, 6:48675–48687, 2018.

D. P. KINGMA et J. BA : Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

J. KOMOROWSKI, G. KURZEJAMSKI et G. SARWAS : Deepball : Deep neural-network ball detector. *arXiv preprint arXiv :1902.07304*, 2019.

M. KRISTAN, J. MATAS, A. LEONARDIS, T. VOJÍŘ, R. PFLUGFELDER, G. FERNANDEZ, G. NEBEHAY, F. PORIKLI et L. ČEHOVIN : A novel performance evaluation methodology for single-target trackers. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2137–2155, 2016.

P. KUMAR, A. DICK et T. S. SHENG : Real time target tracking with pan tilt zoom camera. *In 2009 Digital Image Computing : Techniques and Applications*, p. 492–497. IEEE, 2009.

L. LEAL-TAIXÉ, A. MILAN, I. REID, S. ROTH et K. SCHINDLER : Motchallenge 2015 : Towards a benchmark for multi-target tracking. *arXiv preprint arXiv :1504.01942*, 2015.

P. LEE, Y. UH et H. BYUN : Background suppression network for weakly-supervised temporal action localization. *In Proceedings of the AAAI conference on artificial intelligence*, vol. 34, p. 11320–11327, 2020.

C. LIN, C. XU, D. LUO, Y. WANG, Y. TAI, C. WANG, J. LI, F. HUANG et Y. FU : Learning salient boundary feature for anchor-free temporal action localization. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 3320–3329, 2021.

T.-Y. LIN, P. DOLLÁR, R. GIRSHICK, K. HE, B. HARIHARAN et S. BELONGIE : Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2117–2125, 2017a.

T.-Y. LIN, P. GOYAL, R. GIRSHICK, K. HE et P. DOLLÁR : Focal loss for dense object detection. *In Proceedings of the IEEE international conference on computer vision*, p. 2980–2988, 2017b.

T.-Y. LIN, M. MAIRE, S. BELONGIE, J. HAYS, P. PERONA, D. RAMANAN, P. DOLLÁR et C. L. ZITNICK : Microsoft coco : Common objects in context. *In Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, p. 740–755. Springer, 2014.

C. LIU : *Beyond pixels : exploring new representations and applications for motion analysis*. Thèse de doctorat, Massachusetts Institute of Technology, 2009.

J. LIU, X. TONG, W. LI, T. WANG, Y. ZHANG et H. WANG : Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern recognition letters*, 30(2):103–113, 2009.

W. LIU, D. ANGUELOV, D. ERHAN, C. SZEGEDY, S. REED, C.-Y. FU et A. C. BERG : Ssd : Single shot multibox detector. *In Computer Vision–ECCV 2016 : 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, p. 21–37. Springer, 2016.

F. LONG, T. YAO, Z. QIU, X. TIAN, J. LUO et T. MEI : Gaussian temporal awareness networks for action localization. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 344–353, 2019.

W. LUO, J. XING, A. MILAN, X. ZHANG, W. LIU et T.-K. KIM : Multiple object tracking : A literature review. *Artificial intelligence*, 293:103448, 2021.

N. MA, X. ZHANG, H.-T. ZHENG et J. SUN : Shufflenet v2 : Practical guidelines for efficient cnn architecture design. *In Proceedings of the European conference on computer vision (ECCV)*, p. 116–131, 2018.

S. M. MARVASTI-ZADEH, L. CHENG, H. GHANEI-YAKHDAN et S. KASAEI : Deep learning for visual tracking : A comprehensive survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

F. MASSA et R. GIRSHICK : maskrcnn-benchmark : Fast, modular reference implementation of instance segmentation and object detection algorithms in pytorch. *Google Scholar*, 2018.

A. MILAN, L. LEAL-TAIXE, I. REID, S. ROTH et K. SCHINDLER : Mot16 : A benchmark for multi-object tracking, 2016.

A. MILAN, S. H. REZATOFIGHI, A. DICK, I. REID et K. SCHINDLER : Online multi-target tracking using recurrent neural networks. *In Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

A. NEWSON, A. ALMANSA, M. FRADET, Y. GOUSSEAU et P. PÉREZ : Video inpainting of complex scenes. *Siam journal on imaging sciences*, 7(4):1993–2019, 2014.

J. NING, L. ZHANG, D. ZHANG et C. WU : Robust object tracking using joint color-texture histogram. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1245–1263, 2009.

K. NUMMIARO, E. KOLLER-MEIER et L. VAN GOOL : Object tracking with an adaptive color-based particle filter. *In Pattern Recognition : 24th DAGM Symposium Zurich, Switzerland, September 16–18, 2002 Proceedings 24*, p. 353–360. Springer, 2002.

Y. OHNO, J. MIURS et Y. SHIRAI : Tracking players and a ball in soccer games. *In Proceedings. 1999 IEEE/SICE/RSJ. International Conference on Multisensor Fusion and Integration for Intelligent Systems. MFI'99 (Cat. No. 99TH8480)*, p. 147–152. IEEE, 1999.

M. ONDRAŠOVIČ et P. TARÁBEK : Siamese visual object tracking : A survey. *IEEE Access*, 9:110149–110172, 2021.

D. ONEATA, J. VERBEEK et C. SCHMID : Action and event recognition with fisher vectors on a compact feature set. *In Proceedings of the IEEE international conference on computer vision*, p. 1817–1824, 2013.

N. PAPADAKIS, A. BAEZA, I. RIUS, X. ARMANGUE, A. BUGEAU, O. D'HONDT, P. GARGALLO, V. CASELLES et S. SAGÀS : Virtual camera synthesis for soccer game replays. *In 2010 Conference on Visual Media Production*, p. 97–106. IEEE, 2010.

N. PUSTELNIK, H. WENDT, P. ABRY et N. DOBIGEON : Combining local regularity estimation and total variation optimization for scale-free texture segmentation. *IEEE Transactions on Computational Imaging*, 2(4):468–479, 2016.

F. RAMZAN, M. U. KHAN, A. REHMAT, S. IQBAL, T. SABA, A. REHMAN et Z. MEHMOOD : A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of Medical Systems*, 44, 12 2019.

J. REDMON, S. DIVVALA, R. GIRSHICK et A. FARHADI : You only look once : Unified, real-time object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 779–788, 2016.

J. REDMON et A. FARHADI : Yolo9000 : better, faster, stronger. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 7263–7271, 2017.

S. REN, K. HE, R. GIRSHICK et J. SUN : Faster r-cnn : Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

E. RISTANI, F. SOLERA, R. ZOU, R. CUCCHIARA et C. TOMASI : Performance measures and a data set for multi-target, multi-camera tracking. *In Computer Vision–ECCV 2016 Workshops : Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, p. 17–35. Springer, 2016.

F. ROSENBLATT : The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

O. RUSSAKOVSKY, J. DENG, H. SU, J. KRAUSE, S. SATHEESH, S. MA, Z. HUANG, A. KARPATHY, A. KHOSLA, M. BERNSTEIN *et al.* : Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

A. SADEGHIAN, A. ALAHI et S. SAVARESE : Tracking the untrackable : Learning to track multiple cues with long-term dependencies. *In Proceedings of the IEEE international conference on computer vision*, p. 300–311, 2017.

M. SANDLER, A. HOWARD, M. ZHU, A. ZHMOGINOV et L.-C. CHEN : Mobilenetv2 : Inverted residuals and linear bottlenecks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 4510–4520, 2018.

J. SHIN, S. KIM, S. KANG, S.-W. LEE, J. PAIK, B. ABIDI et M. ABIDI : Optical flow-based real-time object tracking using non-prior training active feature model. *Real-time imaging*, 11(3):204–218, 2005.

Z. SHOU, D. WANG et S.-F. CHANG : Temporal action localization in untrimmed videos via multi-stage cnns. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 1049–1058, 2016.

A. SHRESTHA et A. MAHMOOD : Review of deep learning algorithms and architectures. *IEEE access*, 7:53040–53065, 2019.

K. SIMONYAN et A. ZISSERMAN : Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

K. SUZUKI : Overview of deep learning in medical imaging. *Radiological physics and technology*, 10(3):257–273, 2017.

M. TAN, R. PANG et Q. V. LE : Efficientdet : Scalable and efficient object detection. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, p. 10781–10790, 2020.

J. VALMADRE, L. BERTINETTO, J. HENRIQUES, A. VEDALDI et P. H. TORR : End-to-end representation learning for correlation filter based tracking. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 2805–2813, 2017.

P. VIOLA et M. JONES : Rapid object detection using a boosted cascade of simple features. *In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1, p. I–I. Ieee, 2001a.

P. VIOLA et M. JONES : Robust real-time object detection. vol. 57, 01 2001b.

X. WAN, J. WANG et S. ZHOU : An online and flexible multi-object tracking framework using long short-term memory. *In Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, p. 1230–1238, 2018.

J. WANG et Y. YAGI : Integrating color and shape-texture features for adaptive real-time object tracking. *IEEE Transactions on Image Processing*, 17(2):235–240, 2008.

Q. WANG, J. GAO, J. XING, M. ZHANG et W. HU : Dcfnet : Discriminant correlation filters network for visual tracking. *arXiv preprint arXiv :1704.04057*, 2017.

W. WANG, Y. YANG, X. WANG, W. WANG et J. LI : Development of convolutional neural network and its application in image classification : a survey. *Optical Engineering*, 58 (4):040901–040901, 2019.

N. WOJKE, A. BEWLEY et D. PAULUS : Simple online and realtime tracking with a deep association metric. *In 2017 IEEE international conference on image processing (ICIP)*, p. 3645–3649. IEEE, 2017.

S. XIE, R. GIRSHICK, P. DOLLÁR, Z. TU et K. HE : Aggregated residual transformations for deep neural networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 1492–1500, 2017.

J. XING, H. AI, L. LIU et S. LAO : Multiple player tracking in sports video : A dual-mode two-way bayesian inference approach with progressive observation modeling. *IEEE Transactions on Image Processing*, 20(6):1652–1667, 2010.

Y. XIONG, Y. ZHAO, L. WANG, D. LIN et X. TANG : A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv :1703.02716*, 2017.

H. XU, A. DAS et K. SAENKO : R-c3d : Region convolutional 3d network for temporal activity detection. *In Proceedings of the IEEE international conference on computer vision*, p. 5783–5792, 2017.

L. YANG, H. PENG, D. ZHANG, J. FU et J. HAN : Revisiting anchor mechanisms for temporal action localizationđ. *IEEE Transactions on Image Processing*, 29:8535–8548, 2020.

A. YILMAZ, O. JAVED et M. SHAH : Object tracking : A survey. *Acm computing surveys (CSUR)*, 38(4):13–es, 2006.

Y. YOON, H. HWANG, Y. CHOI, M. JOO, H. OH, I. PARK, K.-H. LEE et J.-H. HWANG : Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning. *IEEE Access*, 7:56564–56576, 2019.

F. YU, W. LI, Q. LI, Y. LIU, X. SHI et J. YAN : Poi : Multiple object tracking with high performance detection and appearance feature. *In Computer Vision–ECCV 2016 Workshops : Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*, p. 36–42. Springer, 2016.

L. ZHANG, H. GRAY, X. YE, L. COLLINS et N. ALLINSON : Automatic individual pig detection and tracking in pig farms. *Sensors*, 19(5):1188, 2019.

X. ZHANG, X. ZHOU, M. LIN et J. SUN : Shufflenet : An extremely efficient convolutional neural network for mobile devices. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 6848–6856, 2018.

H. ZHOU, Y. YUAN et C. SHI : Object tracking using sift features and mean shift. *Computer vision and image understanding*, 113(3):345–352, 2009.

Z. ZOU, K. CHEN, Z. SHI, Y. GUO et J. YE : Object detection in 20 years : A survey. *Proceedings of the IEEE*, 2023.