Projet de Recherche (PRe)

Spécialité : Computer Vision / IA
Année scolaire : 2022/2023

Segmentation automatique et mesure de distance parcourue sur un terrain via vidéo

---

Mention de confidentialité
Rapport non-confidentiel

Auteur :

Dembélé Alex

Promotion :

2024

Tuteur ENSTA Paris : NGUYEN Sao mai
Tuteur organisme d'accueil : PERCHET Vianney

Stage éffectué du 15 /05 /2023 au 18 /08 /2023

NOM de l'organisme d'accueil : ENSAE Paris
Adresse : *5 Av. Le Chatelier, 91120 Palaiseau*

Mention de confidentialité

Le document est non confidentiel. Il peut donc être consultable en ligne par tous

Résumé

Ces dernières années, la pratique du sport à haut niveau à radicalement changée. La recherche de données et d'informations sur les performances et stratégies adverses ainsi que sur ses propres performances tient une place au moins aussi importante que l'amélioration des performances individuelles et l'hygiène de vie. C'est particulièrement le cas pour les sports collectifs, comme le handball, où il est possible de se filmer et surtout filmer les équipes adverses lors de leur matchs. De même avec la médiatisation des sports, de nombreuses chaînes de multimédia filment les matchs. Mais peu de particulier peuvent filmer de manière à récupérer des données de manière efficace. Ce rapport, va s'attacher à essayer des techniques de vision par ordinateur et d'intelligence artificielle afin de permettre à des particuliers de filmer un match de handball avec leur téléphone et de collecter des données en temps réel, sans que ces derniers aient à avoir des connaissances particulières. Les techniques proposés seront plus issues du machine learning que du seul domaine de la vision par ordinateur. Le but va être d'apprendre à une machine à apprendre à calculer une homographie ou une opération similaire et ,dans une moindre mesure, à tracker des joueurs de manière robuste en faisant face au problème d'occultation partielle et totale.

Abstract

In recent years, the practice of high-level sports has undergone a radical change. The pursuit of data and information on both opponents' performance and strategies, as well as one's own performance, holds a position of at least equal importance to enhancing individual performance and maintaining a healthy lifestyle. This is particularly true for team sports, such as handball, where it's possible to record not only one's own matches but also those of opposing teams. Similarly, due to the media coverage of sports, numerous multimedia channels capture the matches. However, few individuals are equipped to film in a way that enables to extract data efficiently. This report will focus on experimenting with computer vision techniques and artificial intelligence in order to enable individuals to film a handball match using their phones and gather real-time data, all without requiring specialized knowledge. The techniques proposed will come more from machine learning than solely the field of computer vision. The objective is to teach a machine to calculate a homography or a similar operation, and to a lesser extent, to robustly track players, addressing the challenges of partial and complete occlusion.

KEYWORD : Handball, Single Camera, Player Tracking, Homography,Machine Learning

Remerciements

Je souhaite remercier l'ENSAE pour m'avoir accueilli lors de mon stage et plus particulièrement l' équipe du CREST pour l'ambiance chaleureuse toute au long du stage. Je remercie surtout MR. Vianney PERCHET , mon maître de stage, pour tous ses conseils et idées, et MR. Hugo RICHARD pour le regard qu'il a porté sur mon stage.

Table des matières

Table des figures

Introduction

De nos jours, les entraîneurs sportifs utilisent beaucoup de données de performances pour réaliser leur entraînements, pour sélectionner les joueurs ou détecter les futurs talents. Cependant, la collecte de ces données est souvent difficile, laborieuse ou chère. Par exemple, la plupart des statistiques d'un match de football sont prises à la main. Des entreprises commencent à proposer des services pour collecter des données à l'aide du traitement des vidéos des rencontres et/ou entraînement. Toutefois, les méthodes employées sont souvent coûteuse en hardware (plusieurs caméra haute performance) ou dispositif sur les athlètes. Le but de ce travail est de réussir à limiter ce coût hardware en développant une solution software performante. Ensuite, les dispositifs actuels profitent souvent du fait que les terrains de football ou volley sont simple avec des lignes contrastées. Les techniques qui sont alors utilisés sont difficilement transposable à des terrains non professionnels ou des terrains plus complexes avec des lignes moins contrastée ou des terrains en moins bon état. Les terrains de handball amateurs sont très différents les uns des autres ; les couleurs sont très différentes d'un terrain à l'autre et la taille du terrain dépend de la taille du gymnase (36mx18m) ou (40mx20m). Le terrain peut aussi comporter des lignes d'autres sports comme celles de basket ou badminton. De plus, les tribunes ne sont pas toutes identiques, donc les caméras qui filment les matchs sont placés à des endroits très variés par rapport au terrain.

FIGURE 1 – Différences terrain handball

Puis, la plupart des analyses ne se font pas en temps réel car elles nécessitent une intervention humaines (au moins pour recalibrer le système lorsque ce dernier commence à se perdre). Un véritable challenge est de réussir à faire un système totalement automatique et en temps réel. Cela permettrait de développer de nombreuses applications autour autre que la prise de statistique, par exemple, des logiciels de paris sportifs en temps réel.

L'objectif principal de ce stage de recherche est de mesurer la distance parcourue par des joueurs de handball à l'aide d'une vidéo. Cela doit se faire à partir d'une vidéo de qualité médiocre prise par une seule caméra. De plus, la caméra n'est pas fixe et ne couvre pas tout le terrain. Elle tourne de gauche à droite pour pouvoir filmer les deux côtés du terrain ; et à cela s'ajoute un bruit aléatoire causé par la personne qui filme.
Ce calcul doit se faire en temps réel et donc de manière totalement automatisé.

Au vu de la grande différence entre les terrains, le choix d'entraîner une/des IA à faire le travail à été fait. L'objectif du stage est donc de trouver une méthode permettant d'annoter des données pour entraîner l'IA. Cette méthode doit être capable de faire la recalibration du système en temps réel, ce qui est traditionnellement fait à la main.

Ce travail permettrait d'aider grandement les entraîneurs et particuliers à tirer des données de performances d'un match de handball. En effet, comme notre étude est faite à partir de vidéo simple de qualité médiocre, cela permettrait à n'importe qui possédant un Smartphone de filmer un match et d'en tirer des statistiques, en temps réel ou non.

Mon tuteur Mr. Vianney Perchet et l'autre chercheur, Mr. Hugo Richard, avec qui j'ai travaillé se sont penchés sur l'aspect mathématiques du problème. Ils ont cherché à le formuler mathématiquement puis cherché des méthodes de résolution. Je me suis plus penché sur l'implémentation des algorithmes et leur performances. Enfin, ce sujet peut se décomposer en deux briques distinctes : le tracking joueurs et la mesure de distance. Après une rapide introduction à la théorie du problème, nous ferons une synthèse de l'état de l'art. Puis nous présenterons la partie sur le tracking pour finir sur la partie la plus travaillé, la mesure de distance.

Ce stage a été effectué dans une équipe de statisticien dont la vision par ordinateur n'est pas forcément la spécialité. Ce rapport explore donc d'avantage des techniques de Machine Learning que de vision par ordinateur.

I Théorie et formalisme

Cette partie permet d'introduire les concepts et notations nécessaires à la compréhension des objets étudiés dans ce rapport. Le début s'appuie grandement sur le livre de Hartley et Zisserman, *Multiple View Geometry in Computer Vision*[9].

Il y a des mentions à openCV qui est une biliothèque de Computer Vision très utilisé (Utilisation de la version Python). Elle s'appuie elle-même sur la bibliothèque Numpy pour réaliser des calculs.

I.1 La géométrie projective

I.1.1 Insuffisance de la géométrie Euclidienne

Tout le monde est familier avec la géométrie Euclidienne et ses concepts. Cependant, il ne sont plus suffisant pour la description géométriques des images prises par un appareil où les notions de perspectives et d'horizons entrent en jeu. La principale limitation vient du concept de ligne parallèle. Cela commence à se pressentir lorsqu'on regarde une photo : les lignes parallèles ne le sont plus au sens d'Euclide : elles se croisent en un point (qui peut se trouver en dehors de l'image). Prenons l'exemple d'une ligne chemin de fer vue d'un train, les deux rails sont séparés de la même distance tout au long de la ligne, pourtant sur une photo, on voit les lignes se croiser à l'horizon.

FIGURE 2 – Chemin de fer se croisant à l'horizon

Dans la géométrie Euclidienne, on définit en général les lignes parallèles comme des lignes se croisant à l'infini. Or, cette notion d'infini n'est pas pratique à manipulée, d'autant qu'elle n'a pas de sens sur une photo (l'horizon est à une distance finie de notre position).

I.1.2 Coordonnées homogène

Il est possible de contourner le problème de l'infini en définissant des points idéaux.

Definition 1 (Point idéal) Un point idéal est un point où deux lignes "parallèles" se croisent sur une image. C'est à mettre en relation avec la notion de point de fuite.

Avec l'introduction de ces points idéaux, l'espace Euclidien devient l'espace projectif. Cet espace est une extension de l'espace Euclidien, cela permet de garder la plupart des propriétés avec lesquelles nous sommes famillières : point, ligne, angle, distance ...

Comme nous travaillons dans un nouvel espace, il est nécessaire de définir un nouveau système de coordonnées. Dans le système Euclidien, le plan est assimilé à $\mathbb{R}^2$ et un point par le doublet (x,y).

Definition 2 (Coordonnées homogènes) Un espace Euclidien $\mathbb{R}^n$ est transformé en espace projectif $\mathbb{P}^n$ par le rajout d'une coordonnée. La représentation du point Euclidien (x,y) est (x,y,1) en coordonnées homogènes. $\forall$ k $\in \mathbb{R}^*$, (x,y,1) et (kx,ky,k) représentent le même point.

Les points homogènes sont définit à une classe d'équivalence près. Pour passer d'un point homogène à un point Euclidien, il faut diviser le triplet par la troisième coordonnée, puis prendre les deux premières. Quand la troisième coordonnées vaut 0, la division donne des coordonnées (x,y) "à l'infini". C'est donc par un triplet ayant sa dernière coordonnée à 0 que l'on traite les points à l'infini ! Tous le points sont considérés comme égaux en coordonnées homogènes. Les lignes parallèles se croisent en un point idéal, point qui a sa troisième coordonnée à O

I.1.3 Transformation simple de l'espace projectif

Il est possible de transformer l'espace Euclidien par des opérations matricielles simple sur les coordonnées : translation, rotations etc... Les points situés à l'infini reste à l'infini après la transformation. Pour appliquer une transformation dans un espace projectif, il faut passer en coordonnées homogène puis faire le produit matriciel. Ici, il n'y a aucune raison qu'un point " à l'infini" le soit encore après l'opération, sa troisième coordonnées peut ne plus être 0.

I.2 Formation d'une image

I.2.1 La caméra : un projecteur

On peut modéliser l'action d'une caméra comme une projection. En effet, une caméra transforme un espace 3D en espace 2D. Pour une caméra à sténopé (tous les rayons lumineux passe par un seul trou), la projection est une projection centrale, tous les points de l'espace 3D (monde réel) forment un point sur l'espace 2D (plan image) en traçant un rayon passant par le trou de l'appareil (centre optique). C'est le même principe pour un appareil avec des lentilles mais avec un trajet plus compliqué pour les rayons.

Definition 3 (Matrice de projection) La matrice de projection $\mathcal{P}$ permet de représenter l'action d'une caméra. $\mathcal{P} \in \mathcal{M}_{3,4}(\mathbb{R})$ (x,y,w)$\in \mathbb{P}^2$, (X,Y,Z,1) $\in \mathbb{P}^3$

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \mathcal{P} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$$

Cette matrice $\mathcal{P}$ est décomposable en produit de deux matrices $\mathcal{M}_{int}$ et $\mathcal{M}_{ext}$ qui contiennent les paramètres intrinsèques et extrinsèques de la caméra.

$$\mathcal{P} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} = \begin{pmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La première matrice contient les paramètres intrinsèques tel que la focale ou les distorsions. La deuxième matrice contient les paramètres extrinsèques comme les translations ou rotations de la caméra par rapport à l'origine.

Il est possible de déterminer les paramètres intrinsèques de la caméra. Ce procédé est appelé étalonnage ou calibration. Cela permet de corriger les distorsions liées aux lentilles.

I.2.2 Matrice d'homographie

Il existe une simplification de la matrice de projection, la matrice d'homographie. On l'utilise lorsque la caméra filme un plan (un terrain de sport par exemple). Elle permet de projeter ce plan dans le plan image.

Definition 4 (Matrice d'homographie) Cette matrice permet de modéliser l'actions d'une caméra sur un plan, c'est une opération projective. Elle met en relation deux plan selon la formule : $\mathcal{H} \in \mathcal{M}_{3,3}(\mathbb{R})$

$$\text{s} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Le s permet de signaler que la transformation se fait en coordonnées homogènes. Le dernier coefficient de la matrice vaut 1 pour assurer l'unicité de la transformation. En effet, comme les coordonnées homogènes sont définis à un facteur près,la matrice d'homographie l'est aussi.

Proposition 1 (Inverse matrice Homographie) *Si l'on peut projeter l'image I dans le système de coordonnées de l'image I' grâce à l'homographie $\mathcal{H}$, alors la matrice $\mathcal{H}^{-1}$ permet de projeter l'image I' dans le système de coordonnées de l'image I.*

Proposition 2 (Homographie successive) *Soient $\mathcal{H}_1$ l'homographie permettant de passer l'image I_1 dans le système de coordonnée de l'image I_2 et $\mathcal{H}_2$ celle permettant de passer de I_2 à I_3. Alors, $\mathcal{H}_2\mathcal{H}_1$ permet de passer de I_1 à I_3.*

I.3 Représentation d'une image

Dans ce rapport nous alternerons entre deux mode de représentation d'image.
— I(x,y)
— I(n)
Cela vient du fait que nous utilisons à la fois des bibliothèques python tel qu'openCV ou Numpy pour traiter des images et des fonctions fait mains qui nécessite une autre représentation.

I.3.1 Représentation I(x,y)

Une image peut être représenter par une matrice I $\in \mathcal{M}_{h,w}$ avec h et w la hauteur et largeur en pixel de l'image. Chaque coefficient x,y de la matrice contient un pixel. C'est un scalaire si l'image est en noir et blanc, et pour une image couleur, c'est un vecteur de taille 3 ou 4 (selon l'espace colorimétrique). Cette réprésentation est celle des bibliothèques de python, notamment openCV qui représente une image par une matrice sous forme d'un array de numpy.

I.3.2 Représentation I(n)

Une image peut être représenter par un vecteur de taille N(le nombre de pixel de l'image). Chaque composante n du vecteur est un vecteur comportant les coordonnées homogènes du pixel et son intensité/sa couleur. Cette représentation permet de faire des opérations sur les coordonnées de manière efficace, notamment le calcul d'homographie.

I.4 Application d'une homographie à une image

Il y a une fonction d'openCV qui permet d'appliquer une homographie à une image en fournissant la matrice d'homographie, l'image de base et ses dimensions : **cv2.warpPerspective**()

Cependant,malgré les apparences, ce n'est pas un simple produit matriciel, ce n'est pas non plus une application linéaire. Comme l'homographie agit sur les coordonnées homogènes, la forme adapté d'une image est I(n). Pour appliquer l'homographie, il faut passer en coordonnées homogènes ou mettre une image sous forme I(n), puis il faut les repasser en forme euclidiennes, en divisant par la troisième coordonnées (La linéarité est perdu ici). Les coordonnées obtenue ne sont pas forcément entière et ne tombent pas forcément dans le cadre de l'image, il faut donc une étape d'interpolation et de suppression des indésirable (la linéarité est totalement perdue ici) . Ainsi, l'application d'une homographie H sur une image I est H(I) et non pas HI.

Algorithme 1 (Application homographie standard)

Objectif : Appliquer une homographie à une image
Données : Image I, Homographie H

Algorithme

- Pour chaque pixel de I :
  - déterminer ses nouvelles coordonnées homogènes : X'=HX
  - déterminer ses nouvelles coordonnées euclidiennes : X'=X'/X'[-1]
- Pour chaque nouvelle coordonnées, vérifier qu'elle tombe bien dans le cadre de l'image
- Interpoler les coordonnées valide
- Générer une image noir I'
- Pour toute coordonnées interpolées, mettre le pixel correspondant de I aux nouvelles coordonnées dans I'.

I.5 Espace Couleurs

Il existe plusieurs espaces de représentation qui permettent de décrire une couleur. La couleur de chaque pixel va être représenté par un vecteur de cet espace. Les espaces les plus connues sont les espaces RGB et niveau de gris.

Espace	Vecteur	Description
RGB	(r,g,b) $\in \{0,...,255\}^3$	Chaque couleur est représenté par un triplet rouge, vert, bleu.
Niveau de Gris	I $\in \{0, ..., 255\}$	Image en noir et blanc
XYZ	$(x,y,z)^T$=M $(r,g,b)^T$ M une matrice 3x3 connue	L'espace xyz permet de représenter des couleurs que RGB ne peut pas
HSV	(h,s,v)$\in$ [0,360]x$[0,100]^2$	H est la teinte "longueur d'onde", S est la saturation ou pureté de la couleur, V est la valeur ou intensité lumineuse
Yuv,	(y,u,v) $\in$ [0,N]x$[-\frac{N}{2}, \frac{N}{2}]^2$	Y représente la luminance, u représente le contraste bleu/jaune, v représente le contraste rouge/cyan
CMYK	(c,m,y,k) $\in [0,100]^4$	Espace pour l'impression par synthèse soustractive cyan, magenta, jaune, noir

Il existe des formules pour passer d'un espace à l'autre. Chaque esapce à ses avantages et inconvénients et est plus utiles dans certaines applications que d'autres.

I.6 Détection de ligne

Cette partie reprend et résume le tutoriel d'openCV consacré à la transformé de Hough.

Il existe une méthode rapide pour détecter des lignes dans une image, la transformé de Hough.

Pour commencer, il faut changer le mode de représentation des droites, on passe de cartésien (x, y) à polaire (r, θ).

L'équation d'une droite peut alors s'écrire : (sous réserve d'être définie) :

$$y = (\frac{-cos(\theta)}{sin(\theta)})x + \frac{r}{sin(\theta)}$$

et donc

$$r = xcos(\theta) + ysin(\theta) \quad (1)$$

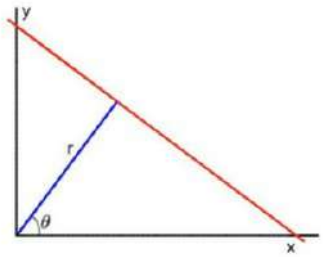

FIGURE 3 – Droite en représentation polaire

Le couple (r, θ) décrit une droite passant par le point(x, y) du plan. Ainsi, pour un point donné (x_0, y_0), il est possible de tracer l'ensemble des droites r_θ, θ passant par ce point $r_\theta = x_0 cos(\theta) + y_0 sin(\theta)$. C'est une sinusoïde. Il faut maintenant déterminer les pixels d'intérêts de l'image, ce sont souvent les contours. Il peuvent être déterminés par une convolution avec un noyau dérivatif ou laplacien. OpenCV dispose aussi de méthodes pour ce faire, tel que la Canny Edge Detection **Canny**() ou **findContours**() d'openCV. Ensuite, pour chaque pixel d'intérêt de l'image,il faut calculer la sinusoïde. Enfin, les points (r, θ) où les sinusoïdes s'intersectent sont des droites passant par les points attachés dont sont issues les sinusoïdes. Par exemple, si 50 sinusoïdes se croisent en un point (r_0, θ_0) alors la droite (r_0, θ_0) passe par les 50 points, dont sont issues les 50 sinusoïdes, Une droite a donc été détectée !

En pratique, l'espace (r, θ) est discrétisé en une matrice $M_{r\theta}$ de dimension la précision de notre discrétisation. Pour chaque pixel d'intérêt de l'image, il faut ajouter 1 au coefficient $M_{r\theta}$ si la droite (r, θ) passe par le point (modulo l'erreur de discrétisation). Une droite est détectée (r, θ) si le coefficient associé de la matrice M dépasse un certain seuil. Cela permet d'avoir un algorithme linéaire en le nombre de pixel d'intérêt et non quadratique ou cubique si l'algorithme était plus direct. Ici ça marche car il faut estimer deux paramètres (r, θ) et donc l'utilisation d'une matrice convient. Il existe deux méthodes openCV pour ce calcul **HoughTransform**() qui calcule des lignes selon cette méthode et qui renvoie r et θ ; **HoughTransformP**() qui utilise cette méthode mais calcul des lignes de longueur finies et qui renvoie les pixels des extrémités de ces lignes.

Il existe des extension de cette technique pour détecter des cercles, ellipses. Cependant la détection d'ellipse requiert la recherche de 5 paramètres ce qui est d'une très grande complexité. Il existe aussi une généralisation de la transformé de Hough qui permet de retrouver n'importe qu'elle forme paramétrée par l'utilisateur, mais elle est aussi d'une complexité très élevé.

II Méthodes actuelles de mesure et tracking vidéo dans le sport

II.1 Enjeux et problèmes

Nous détaillons dans cette parties les différents problèmes importants dans ce domaine.

II.1.1 Occultation

Definition 5 (Occultation) On parle d'occultation lorsque l'objet que l'on cherche à détecter est partiellement ou totalement masquer par un ou plusieurs autres objets qui se trouvent entre lui et la caméra.

Cette occultation peut empêcher le tracking d'un objet sur toutes les images où il apparaît et nécessite d'être capable de réidentifier l'objet après sa perte de localisation.

Dans le cas du sport, les occultations sont en général causé par d'autres joueurs. Ce problème est plus dur à résoudre dans ce cas, car les objets sont de même nature, ce qui est empiré lorsque ce sont deux joueurs de la même équipe portant le même maillot.

Dans notre cas, il y a aussi beaucoup d'occultation totale car la caméra ne film pas tout le terrain à la fois et que tous les joueurs ne sont pas sur le terrain en même temps (il y a de nombreux changement au handball).

II.1.2 Distance dans une image vidéo

Comme une image prise par une caméra est déformé, il est très souvent impossible d'utiliser le système de coordonnée (x,y) des pixels pour mesurer des distances. Il faut alors trouver un nouveau moyen de mesurer les distances. Cela peut se faire en ayant un système de repérage sur nos image(une règle par exemple) ou en transformant l'image dans un nouveau système où l'on peut mesurer les distances.

II.1.3 Mouvement Caméra

Lors du mouvement d'une caméra, la vidéo qu'elle enregistre devient floue. Il devient alors difficile d'analyser correctement les images et d'en tirer des informations. Souvent, le tracking et la détection de terrain ne marche plus pendant le mouvement caméra. Soit le système arrive à poursuivre son analyse, soit le système doit être recalibré pour continuer les calcul.

II.2 Etat de l'art

Un des sports les plus populaires mondialement est le football. Il y a donc beaucoup de travaux prenant comme base ce sport. De plus, il y a de plus gros investisseurs dans ce sport ce qui permet des avancées plus significatives. Nous avons donc travaillé avec des papiers parlant de football en général quand il s'agissait de computer vision dans le sport. Les papiers [8] et [4] font une synthèse des techniques actuelles.

II.2.1 Importance du hardware

Pour traiter les vidéos, il est important d'avoir du matériel de bonne qualité. Mais ce n'est pas le point le plus important. Le nombre de caméra et leurs locations est le facteur le plus important. Il est absolument nécessaire de couvrir l'ensemble du terrain avec le système optique pour ne pas manquer une partie du jeu. Il y a plusieurs choix à faire

- Caméra fixe/mobile : avoir des caméra fixe permet une analyse plus facile des images. En effet, il suffit de configurer le traitement des images dans une seule position ce qui simplifie énormément les calculs. Une caméra mobile permet de couvrir plus de surface avec une seule caméra.
- Champ d'acquisition : une caméra avec un grand champ permet d'acquérir une grande partie du terrain et de limiter la fusion de donnée. Cependant, il n'est pas toujours possible de placer une caméra de telle sorte qu'elle voit tout le terrain sur une image

— Nombre de caméra : Avoir de nombreuses caméras permet de résoudre de nombreux problèmes comme l'occultation. En effet, si un joueur disparaît pour une caméra, il peut toujours être détecté par une autre. La difficulté réside la fusion de donnée et le prix à payer pour les caméras.

De nos jours, la plupart des matchs de football sont filmés par de nombreuses caméras ce qui permet de privilégier l'approche par fusion de données pour le tracking de joueurs. Dans notre cas, nous disposons d'enregistrement d'une unique caméra mobile (que des rotations), dont nous connaissons pas exactement l'emplacement dans le monde réel, qui ne couvre pas tout le terrain en une image.

II.2.2 Tracking des joueurs

Il existe de nombreuses manières de détecter une personne dans une image/vidéo. Il est possible d'utiliser l'histogramme des gradients orientés (HOG) ou la cascade de classifieur de Haar. Mais au vu des innovations actuelles, les meilleures méthodes sont celles utilisant un réseau de neurones convolutif. Un des meilleurs réseau actuel est Yolov8 [6], il est sorti en 2023.

On constate sur l'image 4 que le réseau détecte des joueurs, qu'il les entoure par une bounding box et qu'il

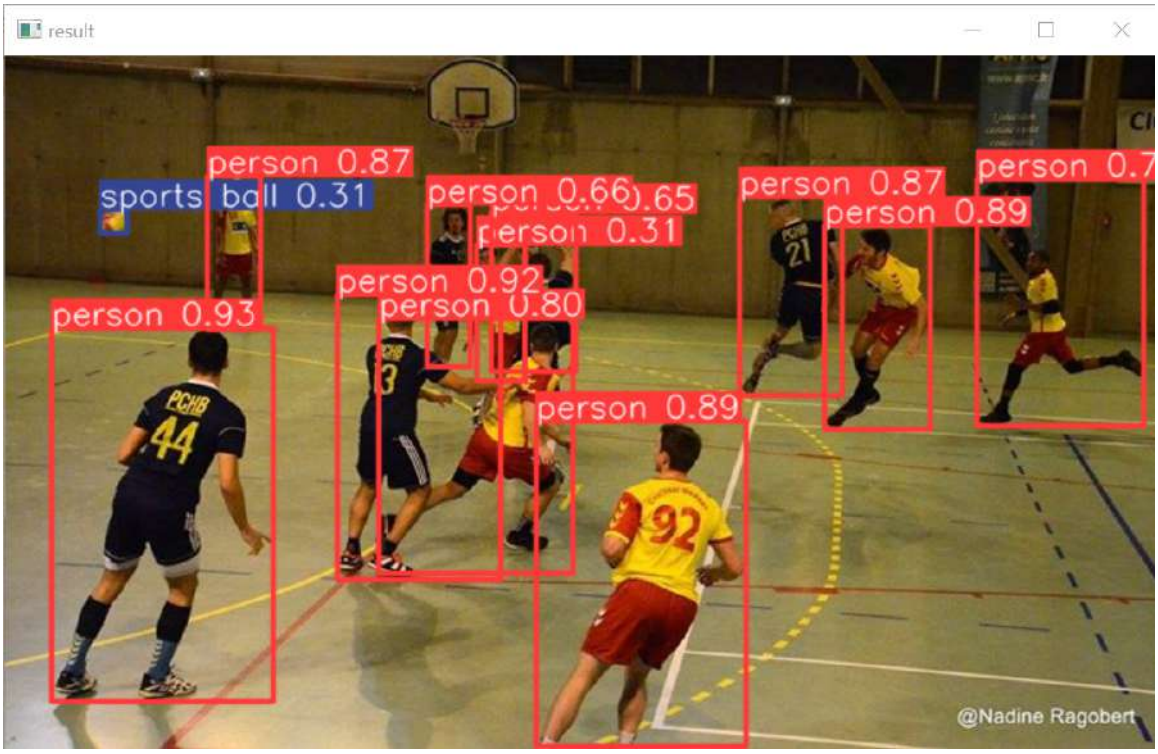

FIGURE 4 – Application Yolov8

donne la qualité de sa prédiction par un score entre 0 et 1 , 1 étant 100% de certitude. De plus, il arrive aussi à détecter la balle, c'est cependant très rare et on voit que le score n'est pas très élevé. La détection de joueur peut être amélioré grâce à de la suppression d'arrière plan (Voir figure 5). Cela consiste à déterminer dans une vidéo une zone qui correspond à l'arrière plan. L'arrière plan est considéré comme les pixels qui ne bougent pas ou peu dans une succession d'image. Une fois l'arrière plan supprimé, les joueurs sont isolés et il est plus facile de les traiter. Une fois un joueur détecté, il faut pouvoir suivre son mouvement pour pouvoir estimer la distance parcourue par ce dernier.

Il existe plusieurs méthodes de tracking tel que le tracking par template,le tracking par feature, ou le tracking par correspondance des bounding box.

II.2.3 Tracking par template

Le but de ce tracking est de suivre un objet dans une vidéo en le reconnaissant à chaque image grâce à un template issue de l'image précédente. A l'image I_{t-1}, t un template de l'objet tracké est extrait sous forme

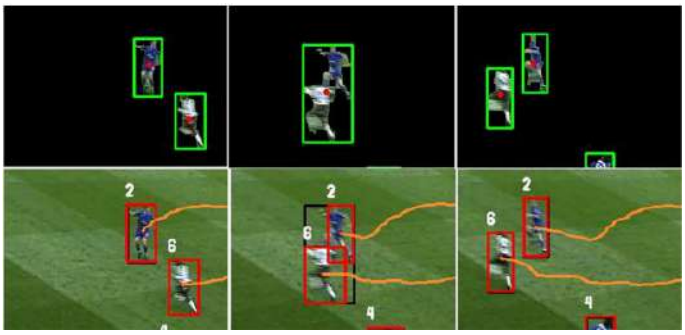

FIGURE 5 – Suppression arrière plan

d'une image de cet objet (bounding box) ou d'un histogramme (couleur ou vitesse) Puis à l'image I_t l'objet est recherché à partir du template en parcourant l'image dans une région à proximité de l'ancienne localisation de l'objet. Utilisé un histogramme est plus robuste qu'une image car il sera moins sensible au bruit.

II.2.4 Tracking par feature

Le tracking par feature reprend le même principe que le traking par template sauf qu'au lieu des templates, des feature sont utilisés, la plupart du temps se sont des points d'intérêts détectés par une méthode SIFT. Cette méthode est plus robuste car les features sont peu sensible au bruit. Le papier [7] propose une méthode de tracking utilisant du tracking par feature et s'appuyant sur l'algorithme Hongrois pour identifier les joueurs et les associer aux détections précédentes. Il est possible de rajouter un filtre de Kalman pour robustifier le tracking par feature et de se servir du flow optique, mais le modèle dynamique des mouvements d'un joueur de handball n'est pas évident, il y a de nombreuses collisions entre les joueurs !

II.2.5 Identification Joueur

En général, lorsqu'un joueur sors du terrain au football, c'est définitif, il n'est pas donc pas nécessaire de savoir identifier un joueur à partir de zéro. Si un joueur disparaît trop longtemps, l'identification est faîte à la main. L'identification peut aussi se faire à l'aide d'un réseau de neurones. Il faut pour cela extraire les box des joueurs de chaque image de la vidéo pour faire une base de données. Puis, il faut annoter ces images afin d'entraîner de manière supervisée un réseau de neurone. Cependant, ces démarches ne permettent pas de faire du temps réel.

Les méthodes automatiques s'appuient en général sur l'identification des numéros [2].

II.2.6 Repérage sur les terrains de sports

Pour mesurer des distances, peu de terrain de sport disposent de graduations précise sur les terrains qui permettent de s'en servir comme étalon de mesure. (Il faudrait des lignes comme celles du football américain mais en quadrillage et qui soient parfaitement distinguable les unes des autres). De plus, la détection de ligne est plus aisé sur un terrain de football ou de hockey sur glace des stades professionnels où il n'y a que des lignes du sport en question et de fort contraste avec le reste.

Certains papiers essaient de reconnaître des formes tel que les lignes de bord du terrain ou les cercles de centre de terrain. Cela peut se faire avec la transformée de Hough mais aussi par du machine Learning en optimisant une certaine fonction qui décrit les forme recherché. La plupart des méthodes proposés s'appuient sur des méthode de deep Learning pour apprendre à une machine à reconnaître et situer un terrain. Cependant, la méthode requiert en général une connaissance préalable du terrain ou un modèle ce qui empêche défaire du temps réel. Par exemple, le papier [5] propose une méthode s'appuyant sur des SVM et du branch and bound pour estimer les points idéaux correspondant à la caméra.

II.2.7 Transformation de l'image

Ainsi, la plupart des méthodes cherchant à obtenir un système de graduation pour faire des mesures de distance réalisent une transformation de l'image prise par une caméra. Le but est de faire coïncider l'image acquise par la caméra avec une image standard sur laquelle on connaît les distances avec les pixels. Cela se fait grâce à une homographie (voir la figure 6). Quasiment toutes les entreprises et papiers de recherche utilisent l'homographie pour faire leurs transformations et localiser des joueurs. Néanmoins, certaines personnes utilisent la matrice de projection pour tenir compte des aspects 3D de la situation.

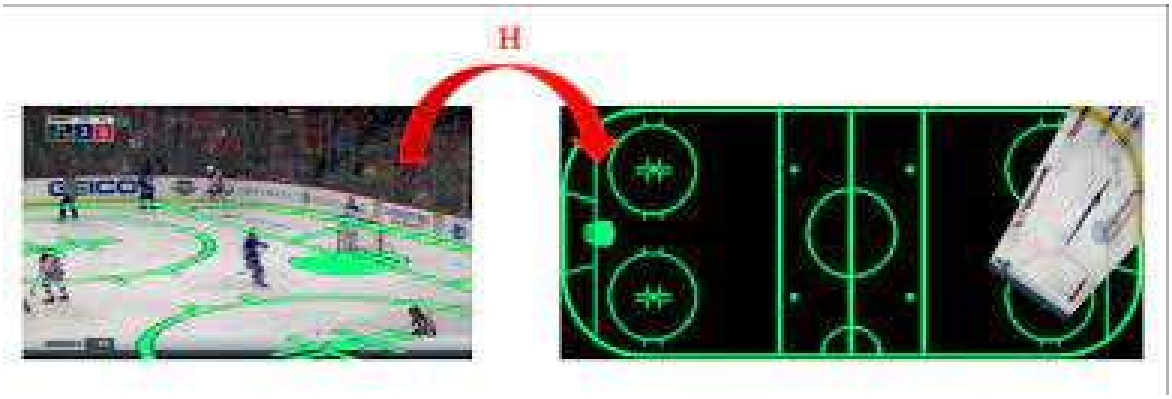

FIGURE 6 – Exemple d'homographie

II.2.8 Calcul d'homographie entre deux images

Pour calculer une matrice d'homographie, il suffit de connaître 4 paires de points de correspondance entre I et I'. En effet, 1 paire de point de correspondance donne 3 équations :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \wedge \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Cependant, une combinaison linéaire de ces équation donne $0 = 0$, donc en réalité, il n'y a que deux équations.(Normal on travail dans un plan) La matrice H ayant 8 coefficient, il faut 8 équations pour les déterminer, d'où les 4 paires de points. La méthode **findHomography**() d'openCV permet de calculer l'homographie à partir des paires de point

Cette méthode requiert d'avoir des positions très précises des points de correspondances. Or il se peut que du bruit de mesure s'ajoute ou que des points soient mal détecté.(Voir partie sur le calcul automatique). De plus, si les points sont très rapprochés l'homographie sera correct dans la zone à l'intérieur du quadrilatère formé par les 4 points, mais sera erronée dans les zones loin de ces 4 points. Enfin, cette méthode ne marche pas si 3 points ou plus sont alignés dans une des images.

Il existe d'autres méthodes qui permettent d'utiliser plus de points de correspondances. Elles se fondent sur la minimisation d'une fonction de coût. Tout la difficulté réside dans le fait de trouver une bonne fonction de coût.

Voici une méthode décrite dans le livre de Hartley et Zisserman [9] :

Etant donnée 1 paire de point de correspondance, on peut former les équations suivantes :

$$\text{x'}=\frac{h_{11}\text{x}+h_{12}\text{y}+h_{13}}{\text{h}_{31}\text{x}+h_{32}\text{y}+h_{33}}$$
$$\text{y'}=\frac{h_{21}\text{x}+h_{22}\text{y}+h_{23}}{\text{h}_{31}\text{x}+h_{32}\text{y}+h_{33}}$$

On a divisé notre résultat du produit matriciel pour obtenir le vecteur ayant sa dernière coordonnée à 1.

En réarrangeant les termes on obtient le système

$$\begin{aligned} \text{x'}(h_{31}\text{x}+h_{32}\text{y}+h_{33}) &= h_{11}\text{x}+h_{12}\text{y}+h_{13} \\ \text{y'}(h_{31}\text{x}+h_{32}\text{y}+h_{33}) &= h_{21}\text{x}+h_{22}\text{y}+h_{23} \end{aligned}$$

On peut écrire ce système matriciellement.

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2}$$

En écrivant ce système pour n points, on obtient une matrice A de taille 2n*9. On peut chercher à résoudre ce système en minimisant $\|Ah\|$ En rajoutant la contrainte pour ne pas obtenir le vecteur nul, on tombe sur un problème de minimisation des moindres carrés.

Probleme 1 (Homographie par moindre carre) $\begin{cases} \textbf{Objectif} : min_h(\|Ah\|) \\ tq : \|h\| = 1 \end{cases}$

La solution de ce problème est le vecteur propre associé la valeur propre minimale de A^TA

II.2.9 Calcul automatique d'homographie

Lors du traitement de vidéos, il est impensable de calculer les homographies à la main en indiquant les points de correspondances sur chaque image. Ce qui est fait en général, c'est l'annotation manuelle de la première image, puis un algorithme calcul automatiquement des points de correspondances entre les images successives de la vidéo. Il peut ainsi déterminer les homographies entre les images de la vidéo . Cependant, comme les calculs automatiques d'homographies perdent en qualité au fur et à mesure de l'avancement, il est souvent nécessaire de recalibrer le système en reannotant des images à la main. Il y a un inconvénient majeur à ce procédé. Il n'est pas possible de faire de temps réel car il y a une intervention humaine.

Le calcul automatique des points de correspondances se fait en général avec un algorithme SIFT ou un de ses dérivés.

II.3 La domination de l'intelligence artificielle

En computer vision, il y a souvent un choix à faire entre utiliser les méthodes traditionnelles qui sont moins coûteuse mais un peu moins performantes et des techniques liés aux deep-learning qui sont plus coûteuses mais plus performantes. Ces derniers années, l'écart de performances est en train de se creuser et l'utilisation d'IA est de moins en moins coûteuse. Elle sont donc de plus en plus utilisés dans ce domaine. Une petite comparaison des performances a été réalisé dans [8]

III Tracking

Cette partie a été peu abordé par manque de temps et ayant été considéré comme moins intéressante (Les techniques actuelles doivent fonctionner dans notre cas)

Le tracking utilisé ici est un tracking s'appuyant sur la détection de joueur par un réseau Yolov8 (nano). Pour chaque image de la vidéo, le réseau renvoie les boundings box des joueurs qu'il a détecté. Ces dernières sont alors utilisés pour réaliser le tracking. Il faut donc réussir à reconnaître le joueur qui se trouve à l'intérieur de chaque bounding box. Cela peut se faire à l'aide des images précédentes, mais pas dans le cas d'occlusion. D'où la nécessité d'utiliser une identification robuste.

III.1 Utilisation du tracking par feature

Il est possible de suivre un joueur sans forcement l'avoir identifier. En effet, plusieurs bounding box représentant le même joueur ne donnent par forcement accès aux mêmes caractéristiques. Par exemple, le numéro de maillot n'apparaît pas sur toutes les images. Il peut aussi apparaître deux personnes à l'intérieur de la même bounding box. L'identification d'un joueur se fait donc sur plusieurs images successives.
Le tracking par feature à été utilisé afin de suivre les box sans forcement avoir identifier les joueurs. Dans chaque bounding box, des caractéristiques sont reconnues afin d'identifier le joueur. Il ne suffit pas de trouver la bounding box la plus proche de l'actuelle dans l'image précédente. En effet, au handball, il y a de nombreuses collisions entre les joueurs.

Les features utilisés sont des points clés détectés à l'intérieur des boundings box. Le détecteur/descripteur utilisé et le KAZE [1].

Sur la figure 7 on constate que l'algorithme KAZE arrive à matcher des points clés dans deux images consécutives de la vidéo, ce qui permet de faire du tracking. Les points clés considérés sont uniquement ceux appartenant à une box.

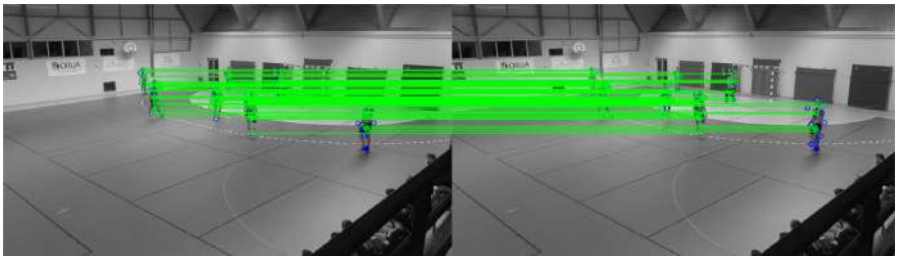

FIGURE 7 – Tracking par feature

III.2 Identification d'un joueur

Comme le processus doit fonctionner en temps réel, le programme ne connaît pas les joueurs en avance et il n'est donc pas possible d'entraîner un réseau à apprendre les caractéristiques des joueurs présent dans la vidéo. Il va falloir que le programme apprenne au vol des caractéristiques des joueurs. Il s'agit donc de trouver de bons descripteurs.

Voici les descripteurs utilisés afin de d'identifier un joueur :

— Numéro de maillot
— Histogramme des couleurs de la bounding box
— Position et mouvement

Le numéro de maillot (s'il est correctement détecté) permet d'identifier un joueur de manière très fiable. Les numéros sont lus avec un réseau de neurones prenant en entrée l'image à l'intérieur d'une bounding box.

Cependant, le numéro n'est pas toujours lisible à l'image ou partiellement occulté (un 16 peut être reconnu comme un 6 !). De plus, deux joueurs peuvent porter le même numéro (un de chaque équipe) (Voir figure 8). Pire, le numéro détecté dans une bounding box peut être celui d'un joueur qui n'est pas celui à qui se rattache la bounding box. Il est donc nécessaire de lire plusieurs fois le numéro de maillot sur une succession d'image. Il faut aussi utiliser d'autres descripteurs dans le cas où le numéro n'est jamais lu.

FIGURE 8 – Deux joueurs dans même box

L'histogramme des couleurs permet de compléter le numéro de maillot. Il permet notamment de déterminer l'équipe à qui appartient le joueur. Cela dépend cependant fortement des couleurs de maillot, du contraste avec le terrain, et de la différence de couleur entre les maillots des équipes. Il est en général plus facile de différencier les maillots de gardien.

En théorie les numéros de maillots et l'histogramme des couleurs sont suffisants pour identifier des joueurs. Pour robustifier le procédé, il est possible d'ajouter des descripteurs comme la position des joueurs. En effet, il y a des postes au handball, les joueurs bougent sur le terrain, mais ils sont plus susceptibles d'occuper certaines zones du terrain. Il est possible d'utiliser cela avec des lois de probabilités par zone pour identifier un joueur. Les joueurs évoluant au bord du terrain quittent sont ceux qui sont le plus susceptibles de disparaître de l'image. Utilisés leur positions permet de rapidement les identifier.

III.3 Algorithme prévisionnel

Algorithme 2 (Tracking de joueur)

Objectif : Tracker les joueurs d emanière robuste

Algorithme

- Pour chaque image I
  - Prédire les bounding box avec Yolov8
  - Utiliser le tracking par feature pour tracker les bounding box
  - Pour chaque bounding box trackée
    - Lire le numéro du maillot si possible
    - Tracer l'histogramme des couleurs
    - Calcul de l'identifiant de la box
    - Si plusieurs identification indique le même joueur , alors identifier la box comme étant celle du joueur X
  - Si des joueurs ont été identifié, mettre à jour leur position comme étant le centre de la bounding box

IV Mesure par détection de ligne

L'idée de cette partie est de détecter les lignes du terrain afin de s'en servir comme étalon de mesure

IV.1 Utilisation de la transformée de Hough

Pour pouvoir utilisé la transformée de Hough, il faut commencer par obtenir les contours de l'image. J'ai utilisé la méthode **Canny**(). En jouant sur les paramètres, il est possible d'avoir plus ou moins de contour. Le problème rencontré est qu'il y a de nombreuses lignes sur le terrain et donc la méthode a du mal à toutes les détectés. Soit peu de contour sont détectés et donc des lignes vont être manquées, soit beaucoup de contour vont être détectés avec du bruit et les lignes seront difficiles à extraire (Voir figure 9). De plus, il y a des contours indésirables sur les images. Ce sont notamment les contours de mur ou fenêtre et de personnes. Ces derniers vont perturber la détection de ligne. Il est possible de supprimer en partie les joueurs grâce aux box renvoyés par le réseau Yolov8 de la détection de joueur.

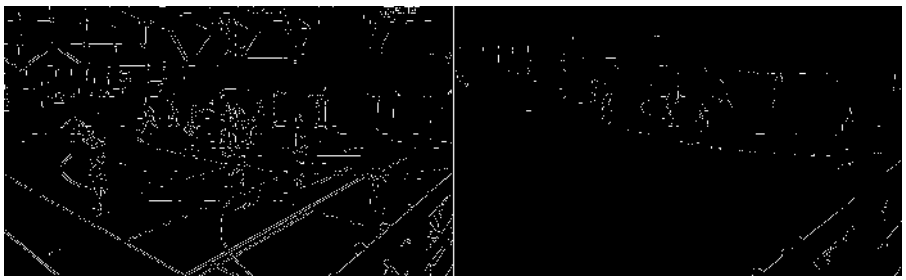

FIGURE 9 – Comparaison de la détection de contour

Une fois la détection de contour réalisée, il est possible d'utiliser la transformée de Hough, j'ai utilisé la transformé **HoughTransformP**() Pour avoir des bout de ligne, cela permet entre autre d'avoir un contour partiel des zones.. On constate sur l'image 10 que les lignes ne sont détectés que partiellement et que beaucoup de ligne détectés ne nous intéressent pas . Il est intéressant de noter que cette méthode est relativement robuste face au image floue qui sont présentes lors d'un mouvement de caméra.

IV.2 Changement d'espace de couleur

Comme la détection de ligne ne marche pas directement, il est possible d'essayer d'utiliser les contraste de couleur du terrain pour la détection de contour . Cela peut se faire en changeant d'espace de couleur. L'espace de couleur qui est le plus adapté en théorie est le HSV, car il est possible de choisir la teinte de couleur ciblée. Cependant, lors des tests, ce n'est pas celui qui s'est montré le plus efficace. Je suis d'abord passer

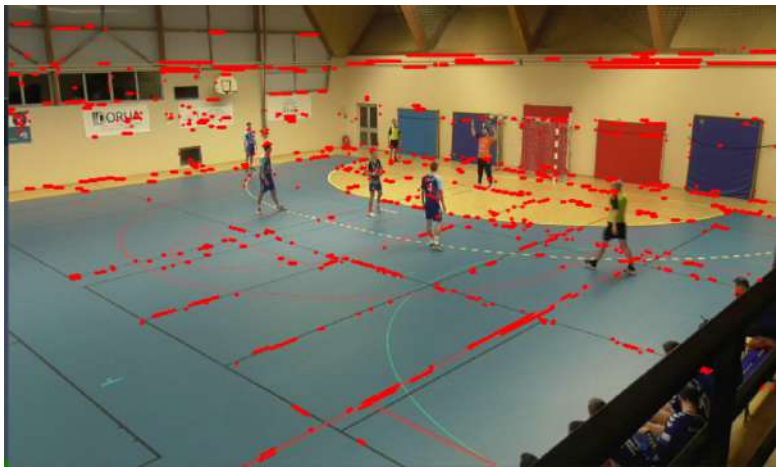

FIGURE 10 – Détection Partielle de ligne

en CYMK pour ne sélectionner que la composante jaune ddes pixels. Mais cela conservait beaucoup de pixel indésirables. Ensuite, j'ai testé de filtret les composantes RGB des couleurs. J'ai passé en noir tous les pixels dont la composante rouge dépasse 160.Ensuite, j'ai supprimé le haut de l'image car j'ai constaté que le terrain ne s'y trouvait jamais. Puis j'ai appliqué la détection de ligne. Comme les images du terrain que j'analysai était majoritairement bleu, alors seul les pixels jaunes/orange ont été conservés. Cela m'a permis d'avoir de meilleurs contours (Voir figure 11)

FIGURE 11 – Détection de ligne avec contour jaune

IV.3 Utilisation de réseau de segmentation

Comme les méthodes précédentes ne permettent pas d'extraire correctement les lignes, l'utilisation d'IA est naturellement venue à l'esprit. J'ai donc annoté des datasets de 200 images pour faire de la segmentation de lignes. Je n'ai annoté que 200 images, car le but est de faire une preuve de concept. Par ailleurs, les seuls lignes qui m'intéressent sont les lignes de handball. On constate que le réseau à du mal à prédire un masque précis des lignes qui sont assez fines(Voir figure 12). Cela vient de l'annotation, la grande majorité des logiciels d'annotation (dont celui que j'ai utilisé) permettent de détourer des polygones pleins mais pas des contours.

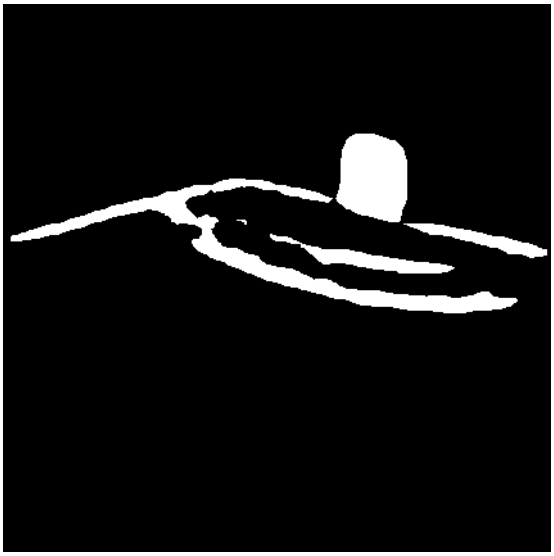

FIGURE 12 – Segmentation des lignes

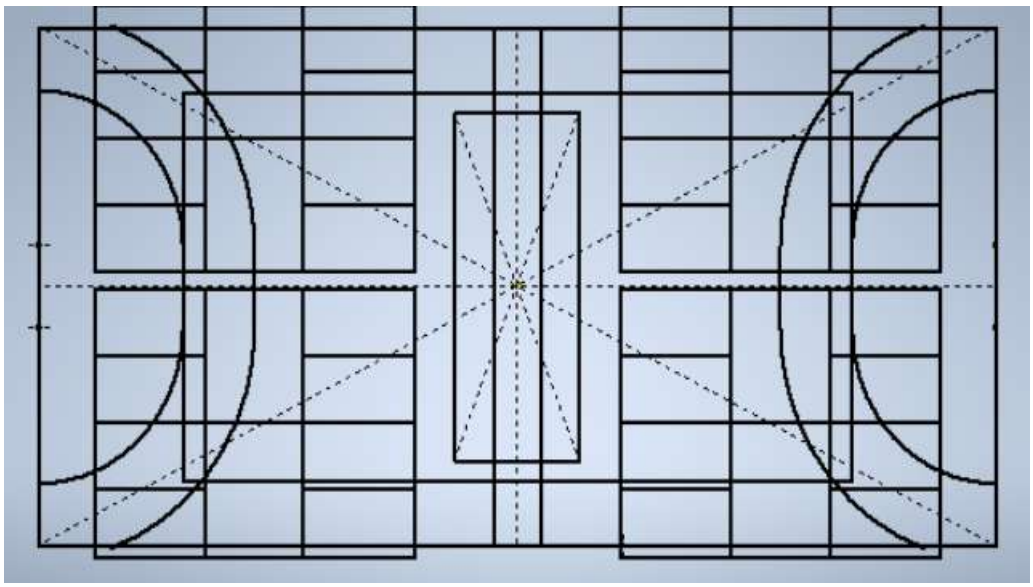

FIGURE 13 – Terrain standard : Jesse Owens

V Calcul d'homographie par optimisation naïve

Notre terrain n'est pas aussi simple qu'un terrain de foot ou de hockey, il contient de nombreuse ligne autre que celle dédiées au handball. Il y a donc beaucoup de bruit. Ainsi la détection de ligne est peu performante même si on essaie de se placer dans un espace de couleur optimal.
Nous avons alors décidé de travailler avec un modèle de terrain dont les dimensions sont connues. Nous avons mesurés toutes lignes (y compris celle externe au handball) pour construire une image 2D de notre terrain. Appelons le résultat "terrain standard". On voit les lignes de handball, mais aussi certaines de badminton et de basket. Nous avons modélisé le terrain du gymnase Jesse Owens de Palaiseau, en France. Nous travaillons donc ,par la suite, avec des vidéos issues de matchs dans ce gymnase . (Voir figure 13)

V.1 Formulation du problème

V.1.1 Calcul d'homographie sans point de correspondance

Le but du jeu est de projeter la première image de la vidéo sur notre terrain standard de manière automatique. Nous pouvons le voir comme un problème de matching : faire correspondre les pixels ou caractéristiques des deux terrains. La première approche à été de le formuler sous forme d'un problème d'optimisation.

Probleme 2 (Homographie sous forme d'optimisation)

Objectif : Trouver $\mathcal{H}$

$$\mathcal{H} = min_H(\mathcal{L}(I' - H(I))) \tag{3}$$

— On peut inverser les rôles de I et I' en prenant H^{-1}
— $\mathcal{L}$ est une fonction de coût (on l'appellera Loss)

Pour que le matching fonctionne correctement, il faut trouver une bonne Loss $\mathcal{L}$. De plus, cette Loss doit essayer d'avoir certaine propriété pour que les algorithmes de minimisations puissent fonctionner. Ainsi, le vrai premier problème à résoudre est :

Probleme 3 (Trouver une bonne Loss)

Objectif : Trouver $\mathcal{L}$ tq

$$\mathcal{H} = min_H(\mathcal{L}(I' - H(I)))$$

— Match correctement les pixels ou des caractéristiques
— Si possible $\mathcal{L}$ soit convexe
— Si possible $\mathcal{L}$ soit $\mathcal{C}^0$ voir $\mathcal{C}^1$

V.1.2 Ajout de contrainte ? !

Nos images ont certaines caractéristiques qu'ils pourraient être judicieux d'exploiter pour améliorer le problème d'optimisation en le transformant en optimisation sous contrainte. En effet, sur toutes les images, les joueurs ont forcément leurs pieds sur le terrain. Les coachs n'ont pas leur pieds sur le terrain, mais leur pieds n'apparaissent pas sur la vidéo, le bas de leur bounding box est quant à lui dans le terrain. De même les arbitres sont tout le temps dans le terrain ou sur les lignes de bord. Rajouter une contrainte prenant en compte cette caractéristique permettrait d'exclure certain minima triviaux des loss. Cependant, ils est aussi possible d'inclure certaines de ses contraintes dans les loss afin de ne pas avoir à calculer des contraintes. Par exemple, en pénalisant dans celle-ci la distance entre le bas des boundings box et le terrain, on tient compte de la contrainte mentionnée ci-dessus. Il faudrait tester laquelle des deux approches est la meilleure (non testé)

V.2 Test de Loss

Je détaille ici certaines Loss qui ont été testés.

V.2.1 Loss avec ligne

Une première idée de Loss est d'utiliser la détection de ligne déjà implémentée. Le but est de faire matcher les lignes détectées et celle du terrain standard. Afin de comparer les deux images, il faut les passer en noir et blanc. Pour la détection de ligne, toutes l'image est en noir, sauf les pixels détectés comme appartenant à une ligne (ou à un contour). Pour le terrain standard, tous les pixels sont noirs, sauf ceux appartenant à une ligne. La Loss revient à tester si un pixel qui est blancs sur une des images l'est aussi sur l'autre. Appelons T le terrain standard et I l'image de détection de ligne.

On intersecte les deux images et comme on cherche une fonction à minimiser on prends l'inverse :

$$\mathcal{L} = \frac{1}{1 + \sum_{i,j} \mathbb{1}_{H(T(i,j))>0} \mathbb{1}_{I(i,j)>0}} \tag{4}$$

Le problème de cette première Loss est qu'elle n'est pas convexe et pas dérivable, l'algorithme de minimisation de scipy va avoir du mal à procéder.

Il est possible d'améliorer les propriétés tout en gardant le même principe de Loss, en essayant d'avoir une Loss quadratique.

$$\mathcal{L} = \sum_{i,j} \mathbb{1}_{H(T(i,j))>0} (H(T(i,j)) - I(i,j))^2 \tag{5}$$

Cette Loss n'est tout de même pas convenable car les images sont très discontinues (même si l'on les prétraite avec un filtre Gaussien) et constante par morceau. L'algorithme va donc avoir du mal à les traiter. De plus, il y a un cas trivial ou la loss vaut 0 : tous les pixels sont envoyés en dehors du cadre de l'image.

V.2.2 Loss comparaison couleur

Pour résoudre en partie le problème de discontinuité des images, on traite les images brutes. Il s'agit ici de comparer la couleur de chaque pixel d'un terrain standard colorié (Voir figure 14) et de l'image de la vidéo. De plus, on veut une Loss différentiable selon l'homographie. Or l'homographie peut renvoyer des pixels à coordonnées non entières. On utilise donc un noyau gaussien pour pouvoir comparer les pixels. Le détail de cette Loss se trouve dans le document joint "sport", il a été écrit par Hugo Richard.

FIGURE 14 – Terrain Standard couleur

V.2.3 Loss distance couleur et distance terrain

Au fil des tests de différentes Loss, nous avons constatés l'importance de deux aspects. Il faut tenir compte des pixels tombant à l'exérieur de l'image de base et il faut sommer sur les pixels de l'image vidéo transformée et donc de supprimer l'étape d'interpolation ce qui rend les images discontinues (Voir figure 15)

Cette loss se compose de deux termes. L'un sert à pénaliser les pixels tombant à l'extérieur du terrain. L'autre à vérifier si les couleurs correspondent bien. Ici, I correspond à l'image issue de la vidéo, sans subir de transformation. Appelons I' l'image à laquelle on a appliqué l'homographie, I'=H(I). Premier terme :

$$\mathcal{L}_1 = \sum_{i,j \in I'} \|(T(i,j)_{couleur} - I'(i,j)_{couleur}\|).[1 - dist((i,j), terrain)]$$

C'est la somme,sur tous les pixels de I', de l'écart de couleur entre les pixels du terrain standard et de l'image homographiée, pondéréé par la distance de ces pixels à l'intérieur du terrain(le terrain est un rectangle et non un point). Ce terme permet de vérifier que les couleurs correspondent bien au niveau du terrain.

Deuxième terme :

$$\mathcal{L}_2 = \sum_{i,j \in I'} \mathbb{1}_{I'(i,j) estbleu}.dist((i,j), terrain)$$

C'est la somme,sur tous les pixels bleu de I', de la distance de ces pixels au terrain(le terrain est un rectangle et non un point). Ce terme permet de vérifier que l'image homographiée retombe bien dans le terrain standard.

Toutes les fonctions distance dist renvoient théoriquement des valeurs entre 0 et 1 : 0 si confondus et 1 si éloignés à l'infini.

On choisit comme distance $e^{-\|\cdot\|}$ pour 1-dist(.) et $e^{-\frac{1}{\|\cdot\|+1}}$ pour dist(.) . La loss final est

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2 \tag{6}$$

Il faut trouver un bon λ pour que les deux termes soient comparables. Dans notre cas, λ=1000 était un bon choix.

Cette Loss a aussi un défaut, pour les images du milieu de terrain sans que les zones apparaissent, il est impossible de correctement les situer sur le terrain ou de savoir leur bonne taille en largeur.

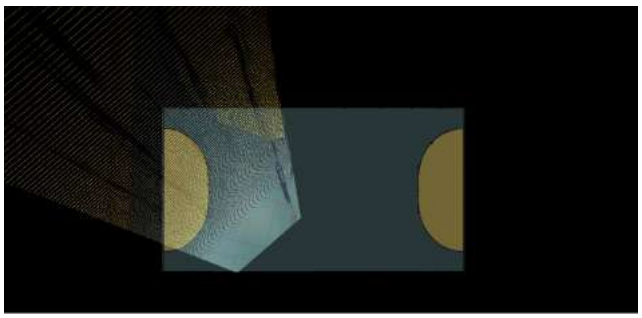

FIGURE 15 – Cadre agrandi et pas d'interpolation

V.2.4 Comparaison des Loss

Pour choisir la bonne Loss, il faut pouvoir les comparer. J'ai annotés 200 images de la vidéo en calculant les homographies correspondantes afin de constituer un début de dataset. J'ai ajouté 100 homographies aléatoires dans ce dataset pour le rendre plus représentatif des stades traversés lors de l'optimisation.

Probleme 4 (Classification par Loss)

Objectif : Comparer les Loss

Soient $(H_i)_{i\in 0...300}$ l'ensemble de nos classes.
Soient $(I_i)_{i\in 0...200}$ l'ensemble de nos images.

$$I_i \in H_j \text{ si } min_{H_k} \mathcal{L}(I_i, H_k) = H_j$$

— Une Loss est adaptée si la classification faîte par cette dernière est la même que l'annotation faîte à la main, donc $I_i \in H_i$

Usuellement, la matrice de confusion C_{ij} est utilisée pour mesurer la qualité d'une classification. C'est une matrice où les lignes et colonnes correspondent aux différentes classes. Chaque coefficient C_{ij} correspond au nombre d'élément de la classe i ayant été classé comme appartenant à la classe j. Par conséquent, une classification parfaite aura une matrice de confusion diagonale.
Cependant, comme ici il y a 200 classes, une matrice ne sera pas facile à analyser visuellement. Les résultats seront donc présentés sous forme d'une courbe. Une image bien classé appartiendra ainsi, à la droite y=x.

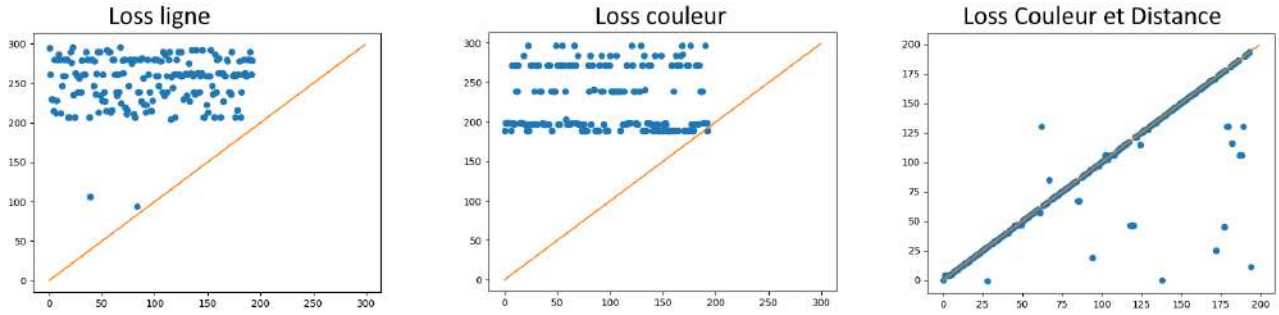

FIGURE 16 – Comparaison des Loss

Resultat 1 (Probleme classification)

(Voir figure 16)

On constate que les deux premières Loss n'arrive pas à correctement classer les images. La plupart des images sont classés dans les homographies générées de manière aléatoire. Pour la première, cela s'explique par le fait que les pixels sont distribués de manière aléatoire et donc qu'il doit y avoir des homographies qui envoient beaucoup de pixel sur les lignes du terrain standard. Pour la deuxième Loss, c'est le même problème. De plus, il y a des classes qui comptabilisent de nombreuses images. Les homographies correspondantes doivent être des transformations qui "exploitent des failles" de la Loss, comme le cas où les pixels sont envoyés hors du cadre de l'image.

Ensuite on constate aussi que la troisième Loss classe plutôt bien les images. De plus, certaines images sont mal classé sur la courbe alors qu'en réalité elles sont bien classés. Par exemple si deux homographies H_i et H_j sont très similaires, les images I_i et I_j peuvent être classé dans la même classe. C'est notamment le cas pour deux images ayant subi la même transformation, l'une représentant un terrain vierge, l'autre représentant

étant un terrain avec des joueurs dessus. Les joueurs sont d'une couleur différente du terrain et donc vont faire augmenter la Loss.

La Loss qui sera utilisée dans la suite sera donc la troisième. C'est la meilleure trouvée jusqu'à l'écriture de ce rapport. Elle n'est cependant par idéale.

V.3 Résolution de l'optimisation

V.3.1 Minimisation avec Scipy

La première méthode pour minimiser nos Loss est d'utiliser la bibliothèque Scipy de python et sa méthode **Scipy.optimize.minimize** . Cette fonction contient plusieurs méthode pour minimiser une fonction, comme CG ou BFGS. La plupart des méthodes nécessitent de fournir la différentielle (et la Hessienne) de la Loss. Or, les Loss ne sont pas forcément $\mathcal{C}^1$. Donc le calcul de la différentielle est souvent impossible. L'algorithme se sert alors de différence finie pour estimer les variations, mais c'est très limité. La résolution sous cette forme a été un échec. L'algorithme trouve des minimums locaux qui correspondent à des cas indésirables. De plus, il y a une forte dépendance à l'initialisation.

V.3.2 Minimisation sous forme de réseau

Une deuxième méthode pour minimiser les Loss est de les mettre sous forme de réseau. Le calcul des dérivés ou la mise sous forme de fonction $\mathcal{C}^1$ des Loss est fastidieux voir impossible. L'idée est de profiter de l'auto-différentiation (autodiff) des bibliothèques de réseaux de neurones. La bibliothèque Pytorch a été utilisé ici. L'autodiff, permet de calculer la dérivé d'une fonction compliqué si cette dernière est composé de fonctions simple dérivable.

Cependant, dans l'application de l'homographie, il y a des étapes qui ne sont pas dérivable comme le test de validité des coordonnées ou la réaffectation des pixels à leurs nouvelles coordonnées. Comme ces opérations ne sont pas supportés par l'autodiff, cette méthode ne peut pas être utilisée. A moins de réussir à régulariser les fonctions sans trop les altérer. (Pas eu le temps de le faire)

V.3.3 Resultat d'optimisation naïve

Voici l'algorithme utiliser pour chercher les homographies sous forme de problème d'optimisation de manière naïve.

Algorithme 3 (Calcul naïf d'homographie)

Objectif : Calculer une homographie permettant de passer d'une image de la vidéo au terrain standard.
Données : Image I, Loss $\mathcal{L}$

Algorithme

- Soit H l'identité
- Minimisation de $\mathcal{L}$:
- Tant que $\mathcal{L}$ n'a pas atteint un minimum
  - Appliquer l'homographie H à l'image I
  - Calculer la Loss correspondante
  - Calculer la différentielle
  - Actualiser les coefficients de H avec la différentielle

J'ai testé cet algorithme avec la bibliothèque scipy.

Resultat 2 (Calcul naïf d'homographie)

En partant de l'identité, l'algorithme converge vers un minimum local qui se trouve à proximité de l'identité. Cela s'explique par le fait que la Loss n'a pas de bonne propriété tel qu'être parfaitement convexe et régulière. Les algorithmes de minimisations ont donc du mal à donner de bons résultats. Pour pallier à ce problème, j'ai essayé d'initialiser le problème à l'homograpie calculé sur l'image précédente de la vidéo (ne résout plus le problème de l'automatisation complète). Cela marche un peu mieux quand il n'y a pas de grande différence entre les deux images successives. Cependant, lorsque les images sont éloignés (notamment lorsque la caméra bouge pour changer de côté du terrain) l'algorithme ne calcule plus de bonne homographie.

En conclusion, l'optimisation naïve ne permet pas de résoudre le problème de l'optimisation totale.

VI Calcul d'homographie avec les paramètres du problème

Jusqu'ici, une matrice d'homographie a été considéré comme avec 8 degré de liberté. Or, en s'intéressant un peu plus à la physique du problème et des mouvements de caméra, il est possible de réduire les degrés de liberté du problème. En effet, la caméra utilisé peut être considéré comme une caméra pan-tilt-zoom (PTZ), c'est-à-dire une caméra qui n'a que 3 degré de liberté :

— Rotation gauche-droite (pan)
— Rotation haut-bas (tilt)
— zoom +/- (zoom)

Les optimisations des parties précédentes n'ont pas marché sûrement car les paramètres optimisés n'étaient pas les réels paramètres du problèmes.

VI.1 Les coefficients de la matrice d'homographie

Intéressons nous de plus près au coefficients de la matrice d'homographie. Pour bien comprendre les effets des coefficients sur les coordonnées, le livre [9] présente des transformations classiques en géométries projective.

Definition 6 (Isométries) Les isométries sont des transformations euclidiennes, elles préservent les longueurs et les angles.
Elles sont données par :

$$\text{M}= \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$$

R : matrice de rotation
T : vecteur de translation $(t_x, t_y)^T$

Definition 7 (Similarité) Les similarité sont la composition d'une isométrie avec une mise à l'échelle isotrope, elles préservent les rapports de longueurs et d'angles.
Elles sont données par :

$$\text{M}= \begin{pmatrix} sR & T \\ 0 & 1 \end{pmatrix}$$

R : matrice de rotation
T : vecteur de translation $(t_x, t_y)^T$ s : un réel représentant la mise à l'échelle isotrope

Definition 8 (Tranformation affine) Les transformations affines sont des transformations linéaires non singulières auxquelles s'ajoutent des translations, elles préservent le parallélisme et le rapport des longueurs sur les même ligne.
Elles sont données par :

$$\text{M}= \begin{pmatrix} A & T \\ 0 & 1 \end{pmatrix}$$

A : matrice 2x2 non singulière
T : vecteur de translation $(t_x, t_y)^T$

La matrice A peut se décomposer en A=R(θ)R(-ϕ)D(λ_1,λ_2)R(ϕ) Avec R des matrices de rotations et D une matrice diagonale.

Definition 9 (Homographie) Une homographie peut se décomposer selon les formes précédentes :

$$\text{H}= \begin{pmatrix} A & T \\ V & 1 \end{pmatrix}$$

A : matrice 2x2 non singulière
T : vecteur de translation $(t_x, t_y)^T$
V : vecteur (v_1, v_2) Le vecteur v permet de transporter les points idéaux à des coordonnées finies et donc de briser le parallélisme entre des lignes.

Ces définitions justifient l'utilisation d'homographie, notamment à cause de la rupture de parallélisme.

VI.2 Détermination des coefficient en fonction des paramètres du problème

VI.2.1 Formulation du problème

Les vidéos dont nous bénéficions sont prises par une caméra qui ne fait pas de zoom +/- au cours de son acquisition , la matrice d'homographie doit donc pouvoir s'exprimer à partir de deux paramètres au cours du mouvement de la caméra. Appelons α et β les paramètres du mouvement, respectivement le pan et le tilt.

Probleme 5 (Homographie à partir de paramètre)

Objectif : Trouver $f_1(\alpha, \beta), ..., f_8(\alpha, \beta)$ tq

$$H = \begin{pmatrix} f_1(\alpha,\beta) & f_2(\alpha,\beta) & f_3(\alpha,\beta) \\ f_4(\alpha,\beta) & f_5(\alpha,\beta) & f_6(\alpha,\beta) \\ f_7(\alpha,\beta) & f_8(\alpha,\beta) & 1 \end{pmatrix} \tag{7}$$

— soit la matrice d'homographie recherchée au cours du mouvement paramétrée par (α, β)
Afin de résoudre le problème ci-dessus, il y a plusieurs méthodes. La première consiste à trouver une formule analytique donnant les coefficients de la matrice en fonction des angles du mouvement. Une telle formule n'a pas été trouvé. Une deuxième consiste à trouver des modèles numériques pour les f_i en analysant des mouvements simples.

VI.2.2 Lien entre les espaces de paramètres

Nous avons deux espaces de paramètres pour faire varier les coefficients de la matrice d'homographie. Le premier correspond directement à celui des coefficients. Il n'est pas le plus représentatif des degrés de liberté du problème. Le deuxième et celui de la définition 9.

Pour pouvoir tracer les courbes empiriques, il faut être capable de passer d'un espace à l'autre. Passer de l'espace de la définition 9 à l'autre et facile, il suffit de faire des produits matriciels en suivant la définition, ce n'est pas le sens qui nous intéresse le plus. Cependant, pour passer de l'espace des coefficients à celui de la définition 9, je n'ai pas trouvé de formule analytique pour tous les paramètres. J'utilise donc une méthode numérique utilisant la méthode **scipy.optimize.root** de scipy qui permet de trouver les zéros d'un système d'équations. Au vue de la définition, seul les paramètres θ, ϕ, λ_1 et λ_2 sont à calculer. Il faut résoudre le système suivant :

$$R(\theta)R(-\phi)D(_1, \lambda_2)R(\phi) = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{8}$$

Il y a une expression analytique pour θ

$$\theta = \arctan\left(\frac{h_{21}-h_{12}}{h_{11}+h_{22}}\right)$$

Puis, on met le système sous la forme

$$R(-\phi)D(_1, \lambda_2)R(\phi) = \begin{bmatrix} A & B \\ B & C \end{bmatrix} = R(-\theta)\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \tag{9}$$

Avec

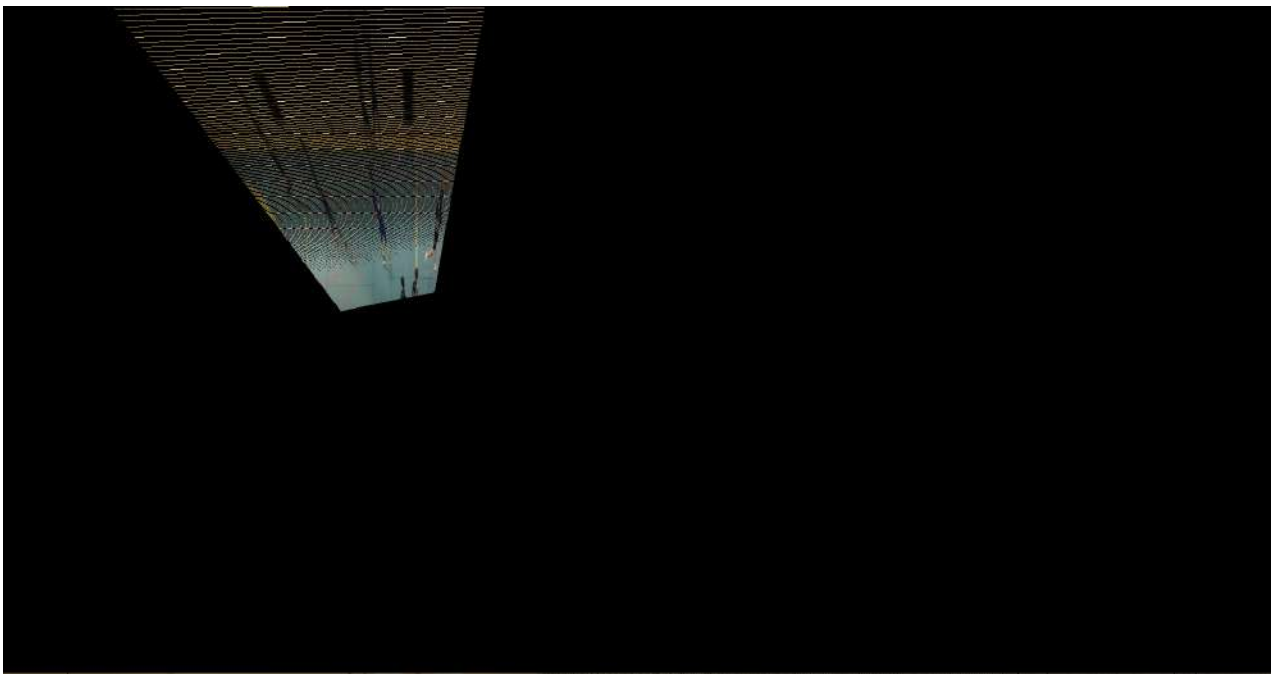

FIGURE 17 – Plage angulaire conique

$$\text{A}=\lambda_1 cos^2(\phi) + \lambda_2 sin^2(\phi)$$
$$\text{B}=(\lambda_2 - \lambda_1) sin(\phi) cos(phi)$$
$$\text{C}=\lambda_2 cos^2(\phi) + \lambda_1 sin^2(\phi)$$

On résout ce système avec la métode root de scipy.

VI.2.3 Resolution par modèle numérique

Pour trouver une formule empirique donnant l'évolution des coefficient de la matrice d'homographie, j'ai commencé par trouvé un extrait de vidéo ou la caméra fait un mouvement de gauche à droite. Puis, j'ai calculé à la main les homographies de toutes les images de l'extrait. J'ai ensuite estimé la variation angulaire de la caméra au cours de son mouvement.

On constate qu'une fois l'homographie appliqué sur l'image, la plage du champ de vison de la caméra apparaît sous forme d'un cône (VOir figure 17). Pour paramétrer mon mouvement, j'ai pris l'angle moitié entre les angles des droites du cône par rapport à la verticale.

Enfin, j'ai tracé les courbes d'évolutions pour les coefficients H_{ij} et ceux de la définitions 9.

Resultat 3 (Paramétrisation simple d'une homographie)

On constate que les courbes sont très bruités (Voir figure 18). Cela vient du fait que la rotation de la caméra n'est pas pur et que je n'utilise pas de formule analytique pour passer des h_{ij} au paramètres de la définition 9. De plus, l'espace de la définitinon 9 est le meilleur car certains paramètres semblent constant au cours du mouvement.

Lorsqu'on modélise les courbes avec des expressions mathématiques et qu'on calcul les homographies avec ces modèles en faisant varier theta, le mouvement de rotation apparaît seulement autour de la plage de modélisation(normal) mais les homographies sont très peu précises.

Une fois les modèles trouvés, il est possible de reprendre les problème d'optimisation mais en minimisant sur moins de paramètres.Les résultats sont meilleurs mais toujours relativement imprécis. Cela vient du fait que les paramètres ne sont toujours pas les bons. Les rotations ne sont pas autour des bon axes !

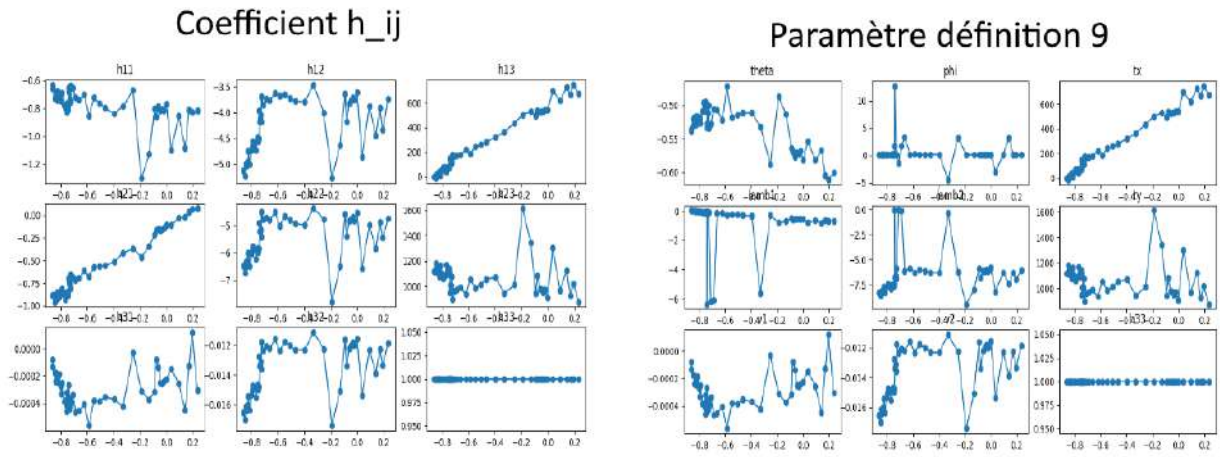

FIGURE 18 – Courbe Empiriques Homographie

Cette méthode n'est donc pas idéale car elle dépend fortement de la précision des modèles et prend du temps pour faire les acquisitions et les annotations à la main. Ce sera fastidieux de faire les modèles à deux paramètres.

VI.3 Paramétrisation de la matrice de projection

Utiliser les degrés de liberté du problème est plus efficace avec la matrice de projection. En effet, celle-ci tient compte de la 3D du problème, notamment le positionnement de la caméra dans l'espace. Cette démarche est proposée dans le papier [3].

VI.3.1 Modéliation de la caméra

On attache un repère au monde réel dont l'origine est le coin en bas à gauche du terrain. On attache aussi un repère à la caméra. Il est possible de modéliser une caméra PTZ comme suit :

$$P = \mathcal{M}_{int} R_p R_t S_C \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \end{pmatrix} \tag{10}$$

— Le vecteur C=(C_x, C_y, C_z) est le positionnement de la caméra dans les coordonnées du monde.
— S_C est la matrice de rotation du repère attaché à la caméra.
— R_p et R_t sont les matrices de rotation de pan et tilt
Voici le problème à résoudre :

Probleme 6 *(Matrice de projection PTZ)*

Objectif : Trouver $\mathcal{M}_{int}$,θ_{pan},θ_{tilt} tq

$$P = \mathcal{M}_{int} R_p R_t S_C \begin{pmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \end{pmatrix} \tag{11}$$

— soit la matrice de projection donnant l'image prise par la caméra

$\mathcal{M}_{int}$ peut-être trouvé en faisant de la calibration caméra.
θ_{pan},θ_{tilt} peuvent être trouvé par un problème d'optimisation ou un réseau de neurones ou en générant plein d'angle de manière uniforme(brute force)

La calibration de la caméra peut se faire avec openCV et sa méthode**calibrateCamera**() Il suffit pour cela de prendre une série de photos d'échiquiers. Donc la matrice $\mathcal{M}_{int}$ n'est à priori pas un paramètre du problème. Elle devient un paramètre si la caméra fait du zoom/dé-zoom en plein match.

VI.3.2 Test du modèle de projection PTZ

Pour tester le modèle précédent, j'ai codé un moteur caméra qui permet, après avoir placé celle-ci dans l'espace (Cx=20, Cy=-3, Cz=4), de faire tourner la caméra avec le clavier. Cela a permis de tester le modèle.

Resultat 4 (Validation modèle PTZ)

Lors des essais du modèle PTZ, j'ai constaté que les angles de pan et tilt était effectivement les angles qui paramétrait majoritairement le mouvement. Cependant, j'ai aussi constaté que l'angle de roll n'est pas constant. Cela s'explique par le fait que la caméra qui prend nos vidéos n'a pas son axe de roll bloqué. Donc comme c'est un humain qui prend les images il va inévitablement un peu tourné selon cet axe. De plus, dans le cas ou des vidéos seraient prises par des particuliers avec leurs smartphone, le roll sera aussi un angle non négligeable. Finalement, il semble impératif de prendre en considération l'angle de roll dans le modèle.

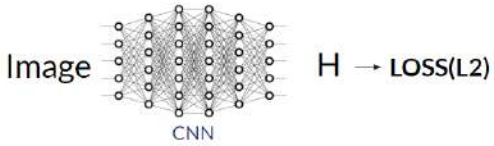

FIGURE 19 – Réseau pour apprentissage supervisé

VII Robustification de l'automatisation totale

VII.1 Principe du réseau

L'utilisation d'un réseau de neurone permet de robustifier la prédiction d'homographie ou de matrice de projection. En effet, comme les terrains et leur couleurs sont très variables, les Loss que nous avons déterminés précédemment ne sont plus efficaces. Le but est d'entraîner le réseau sur un/des terrains où l'annotation avec la Loss marche. Puis, il faudra étendre le réseau sur d'autres terrain par apprentissage successif.

Il faut faire des prédictions avec les réseaux sur des terrains similaires à ceux d'entraînement, en espérant qu'ils arrivent à généraliser le problème. Puis, on construit un dataset avec les prédictions corrects. Ensuite on réentraîne le réseau en incorporant ce nouveau dataset à l'entraînement. Enfin, on recommence les prédictions sur un terrain encore un peu différents.

En répétant cette procédure, on espère que le réseau généralise petit à petit sur tous les types de terrains de handball, quitte à lui indiquer en entrée les couleurs ou la taille du terrain. C'est de l'apprentissage par transfert.(Données accessibles par les particuliers voulant filmer un match)

VII.2 Type de réseau

Deux architectures de réseaux ont été imaginés pour remplir la tâche. Ils se basent tous les deux sur un réseau convolutif (CNN) qui est résilient au bruit et qui permet d'extraire des features des images. Ce CNN peut être un Yolov8, un ImageNet ou consorts.

VII.2.1 Apprentissage supervisé

Le premier type de réseau serai juste une modification de la dernière couche d'un CNN classique. Par exemple, on supprime la dernière couche linéaire qui prédit les classes dans un Yolov8 et on la remplace par une couche qui prédit 8 flottants qui correspondent aux 8 coefficients inconnus de la matrice d'homographie. Puis on utilise une loss classique tel que la lossMSE (norme L_2) pour réalisé l'apprentissage sur une base de donnée que l'on aura constitué avec les parties précédentes.

Il est aussi possible de modifier la dernière couche du réseau pour apprendre les paramètres du modèle de caméra de la partie précédentes. Mais il faudrait sûrement modifier un peu la loss car on annote des homographies et pas des angles.

L'architecture se trouve sur la figure 19 .

VII.2.2 Méthode non-supervisé

Ici on reprend la méthode précédente mais une utilise pas la Loss L_2 On la remplace par une des Loss que l'on a cherchés dans les parties précédentes, mais mise sous forme de réseau de neurones. Le but est de profiter de l'auto-différentiation des bibliothèque python telle que Pytorch. Il faut geler les couches de la Loss pour que les poids ne soient pas mis à jour lors de la backward propagation. Le but du réseau est de prédire 0. Il n'y arrivera jamais, mais ça le forcera à minimiser la Loss. Pour prédire les coefficients de l'homographie (ou les paramètres de la matrice de projection) il faut fournir une image en entré du CNN et du réseau calculant la Loss et faire des boucles d'apprentissage avec cette seule image en entrée. Le réseau va alors calculer les coefficients de la matrice d'homographie de manière non supervisé.

L'architecture du réseau se trouve sur la figure 20 .

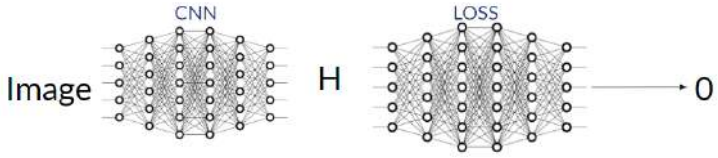

FIGURE 20 – Réseau pour apprentissage non-supervisé

Semaine 1	-Acquisition de vidéo -Recherche sur l'état de l'art -Détection de joueur avc Yolov8
Semaine 2	-Recherche sur l"état de l'art -Rencontre d'un chercheur ayant fait sa thèse en Vision -Test de méthode de détection de ligne avec openCV - Decouverte et compréhension de la matrice d'homographie
Semaine 3	-Annotation d'image pour de la détection de ligne - Entrainement de réseau pour de la détection de ligne - Formulation du problème d'optimisation - Calcul d'homographie
Semaine 4	Annotation d'image pour de la détection de ligne - Entrainement de réseau pour de la détection de ligne - Algorithme pour la minimisation avec scipy - Essai d'une première Loss -Compréhension de la géométrie projective
Semaine 5	-Essai du tracking par feature -Amélioration de l'algorithme de minimisation -Recherche d'une meilleure Loss
Semaine 6	- Recherche d'une Loss régulière -Dérivation et implémentation de la Loss trouvée -Essai de la Loss trouvée
Semaine 7	- Application d'homographie sans utiliser openCV -Amélioration des performances temporelles des algorithmes -Rédaction Rapport
Semaine 8	-Recherche d'une meilleure Loss (Somme de deux termes discriminant -Annotation de 200 images avec leur homographies associés
Semaine 9	- Comparaison des performances des différentes Loss -Annotation d'image pour trouver les points clés d'un terrain. -Entraînement d'un réseau pour trouver les points clés d'un terrain.
Semaine 10	-Essai de mettre une Loss sous forme de réseau de neurone -Reflexion sur l'architecture finale du/des réseaux
Semaine 11	- Recherche de nouvelles techniques dans l'état de l'art
Semaine 12	-Recherche sur la manière de paramétré une homographie
Semaine 13	-Rédaction du rapport -Calcul d'homographie à partir des paramètres du problème
Semaine 14	-Rédaction du rapport -Paramétrisation de la matrice de projection

TABLE 1 – Emploi du temps

Conclusion

Pour conclure, ce stage a permis d'appliquer des techniques de machine-learning en computer vision. Il a eu pour but d'essayer des algorithmes permettant de faire du temps réel. Au vu des contraintes du problème comme les occlusions ou la faible qualité du hardware, les techniques naïves ne fonctionnent pas. C'est pourquoi il faut s'attacher à bien connaître la théorie du problème pour avoir des modèles les plus précis possible. La recherche de Loss est fondamentale dans ce genre de problème de matching car l'optimisation dépend beaucoup de la fonction trouvée. La loss trouvée dans ce rapport n'est pas optimale est peut être largement amélioré ; elle ne marche aussi que dans le cas du terrain pour lequel elle a été formulé. Je n'ai pas eu le temps de beaucoup manipuler la matrice de projection paramétrée, mais ça semble être la piste la plus prometteuse afin de calculer des transformations en temps réel.

Ce rapport n'est pas représentatif de l'ordre dans lequel le travail a été fait. Comme c'était un travail de recherche il y a eu de nombreux aller-retour, pause et moment de réflexion qui font que le chemin était serpentueux.

D'un point de vue personnel, ce stage m'a permis d'apprendre et d'appliquer de nombreuses connaissances. Ne faisant pas la filière math appliqué, ce stage m'a permis d'acquérir des savoirs que je ne verrai peut-être pas en cours(comme des savoirs sur l'optimisation et la formulation de problème), grâce à l'équipe de statisticien dans laquelle j'étais. Ensuite, comme il est difficile de combiner robotique et IA en 3eme année, ce stage m'a permis d'utiliser des outils d'IA que je n'aurai , à priori, pas l'occasion de manipuler l'année prochaine. La réelle plus value de ce stage, ce sont toutes les compétences apprises dans le domaines !

Finalement, bien que le stage n'est pas aboutit sur un système fonctionnel, il pose néanmoins les bases d'un travail de R&D dans ce domaine. Il y a encore de longues heures de travail pour arriver à un produit fini.

Glossaire

— **BoundingBox** : rectangle entourant la prédiction d'un réseau de neurones
— **CNN** : Convolutionnal Neural Network
— **Loss** : Fonction de coût
— **CamraPTZ** : Caméra pouvant zoomer et tourner autour de deux axes de rotation : lacet(pan) et tangage(tilt)
— **roll** : Rotation correspond au roulis
— **SVM** : Support Vector Machine

Références

[1] Pablo Fernández Alcantarilla, Adrien Bartoli et Andrew J. Davison. "KAZE Features". In : *Computer Vision – ECCV 2012*. Sous la dir. d'Andrew Fitzgibbon et al. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 214-227. isbn : 978-3-642-33783-3.

[2] Marco Bertini, Alberto Del Bimbo et Walter Nunziati. "Player Identification in Soccer Videos". In : *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*. MIR '05. Hilton, Singapore : Association for Computing Machinery, 2005, p. 25-32. isbn : 1595932445. doi : 10.1145/1101826.1101833. url : https://doi.org/10.1145/1101826.1101833.

[3] Jianhui Chen et James J. Little. *Sports Camera Calibration via Synthetic Data*. 2018. arXiv : 1810.10658 [cs.CV].

[4] T. D'Orazio et M. Leo. "A review of vision-based systems for soccer video analysis". In : *Pattern Recognition* 43.8 (2010), p. 2911-2926. issn : 0031-3203. doi : https://doi.org/10.1016/j.patcog.2010.03.009. url : https://www.sciencedirect.com/science/article/pii/S0031320310001299.

[5] Namdar Homayounfar, Sanja Fidler et Raquel Urtasun. "Sports Field Localization via Deep Structured Models". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, p. 4012-4020. doi : 10.1109/CVPR.2017.427.

[6] Qiu Jing Jocher Glenn Chaurasia Ayush. *YOLO by Ultralytics, v8.0.0*. url : https://github.com/ultralytics/ultralytics. (accessed : 14/08/2023).

[7] Miran Pobar et Marina Ivasic-Kos. "Active Player Detection in Handball Scenes Based on Activity Measures". In : *Sensors* 20.5 (2020). issn : 1424-8220. doi : 10.3390/s20051475. url : https://www.mdpi.com/1424-8220/20/5/1475.

[8] Pegah Rahimian et Laszlo Toka. "A survey on player and ball tracking methods in soccer and other team sports". In : *Journal of Quantitative Analysis in Sports* 18.1 (2022), p. 35-57. doi : doi:10.1515/jqas-2020-0088. url : https://doi.org/10.1515/jqas-2020-0088.

[9] Andrew Zisserman Richard Hartley. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. isbn : 9780521540513.