Analyse de détections concomitantes pour l'aide à la navigation interactive dans de grandes collections de vidéos

Thierry Malon

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique et Télécommunication

Présentée et soutenue par :

M. THIERRY MALON

le lundi 14 décembre 2020

Titre :

Analyse de détections concomitantes pour l'aide à la navigation interactive dans de grandes collections de vidéos

---

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (IRIT)

Directeur(s) de Thèse :

M. VINCENT CHARVILLAT

MME SYLVIE CHAMBON

CROUZIL ALAIN

Rapporteurs :

MME LAURE TOUGNE, UNIVERSITE LYON 2

MME MICHELE GOUIFFES, UNIVERSITE PARIS-SUD

Membre(s) du jury :

M. WILLIAM PUECH, UNIVERSITE DE MONTPELLIER, Président

M. ALAIN CROUZIL, UNIVERSITE TOULOUSE 3, Membre

MME FLORENCE SEDES, UNIVERSITE TOULOUSE 3, Invité(e)

MME SYLVIE CHAMBON, TOULOUSE INP, Membre

M. VINCENT CHARVILLAT, TOULOUSE INP, Membre

Titre : Analyse de détections concomitantes pour l'aide à la navigation interactive dans de grandes collections de vidéos

Résumé :

Qu'il s'agisse de la caméra d'un smartphone ou d'une caméra de surveillance, les dispositifs d'enregistrement vidéo sont de plus en plus répandus et les quantités de vidéos disponibles ne cessent de croître. Cette surabondance de données disponibles est une épée à double tranchant : plus les vidéos de la scène dont on dispose sont nombreuses et variées, plus elles offrent une information riche et complète, mais plus elles nécessitent de temps pour être traitées. Dans de nombreuses applications, comme par exemple une enquête policière, les premières heures d'investigation sont décisives. Or la visualisation exhaustive de toutes les vidéos par un opérateur humain est une tâche pénible, peu efficace, longue et coûteuse. Cette thèse propose d'explorer plusieurs pistes dans le but d'automatiser la recherche d'information dans des vidéos. Plus précisément, nous cherchons à identifier parmi une collection de vidéos celles issues de caméras présentant un recouvrement partiel de leurs champs de vue. Pour réaliser cette tâche, nous avons également mis en oeuvre une méthode de navigation interactive entre les vidéos. Ainsi, dans ce contexte, trois contributions principales sont apportées.

Tout d'abord, nous proposons une méthode permettant de déterminer quelles vidéos ont leurs champs de vue qui se recouvrent et offrent ainsi différents points de vue d'un même endroit. Pour ce faire, nous détectons la présence d'objets dans les vidéos au cours du temps, les caractérisons par une catégorie et une apparence et regroupons les vidéos qui présentent de façon concomitante des objets de même catégorie aux apparences similaires.

Cependant, il arrive que des objets aient des apparences similaires bien qu'ils soient issus de paires de caméras dont les champs de vue ne se croisent pas, ce qui cause des erreurs dans les groupements de vidéos. Pour y remédier, nous proposons un mécanisme d'apprentissage actif, où le programme peut solliciter l'opérateur sur des cas difficiles pour apprendre à trouver les groupements de caméras qui présentent du recouvrement dans leurs champs de vue. Nous enrichissons également les critères employés en considérant l'arrière-plan en plus des objets apparaissant dans la vidéo. Nous montrons que les groupements de vidéos obtenus par la coopération entre l'homme et la machine sont meilleurs que ceux obtenus via notre approche précédente, entièrement automatisée.

Enfin, nous proposons une méthode de navigation au sein d'un groupe de vidéos observant un même endroit. Depuis une vidéo courante, l'approche proposée permet aux utilisateurs de tracer une trajectoire requête et les redirige vers une autre vidéo dans laquelle cette trajectoire est plus longue, plus détaillée et donc mieux observable. Nous apprenons pour cela des cartes de correspondances entre régions de chaque vidéo et les utilisons pour reformuler la trajectoire tracée par l'utilisateur en son équivalent dans les autres vues.

Afin de valider nos approches, nous identifions plusieurs jeux de données pertinents dans notre contexte. Cependant, le nombre de caméras dont les champs de vue se recouvrent est généralement limité. Nous proposons donc un jeu de données multivues que nous avons annoté et qui contient 25 caméras disposées sur un campus universitaire dont 19 présentent du recouvrement. Les résultats obtenus au travers de nos trois contributions sont encourageants.

Title: Analysis of concomitant detections for interactive navigation in large video collections

Abstract:

Video recording devices are becoming more and more common, and the amount of video available keeps on growing. This huge amount of available data is a double-edged sword: the more videos of the scene are available, the richer and more complete the information, but the longer it takes to process them. In some applications, such as a criminal investigation, the first few hours are crucial, and the exhaustive viewing of all the videos by a human operator is a tedious, unefficient, long and costly task. This thesis proposes to explore several directions in order to automate video analysis processing. For this purpose, we also introduce navigation strategies between different videos. Thus, we propose three main contributions.

First of all, we introduce a method for determining which videos share overlap in their fields of view, and consequently offer different viewpoints of the same location. In order to do so, we detect the objects in the videos over time, describe them by a category and an appearance, and group videos containing concomitant detections of objects of the same category with similar appearances.

However, some objects can have similar appearances even between pairs of cameras with non overlapping fields of view, which leads to errors in the video clusters. To solve this, we propose an active learning mechanism, in which the program can ask the operator for difficult cases to learn finding groups of cameras with overlapping fields of view. We also enrich the criteria used by considering the background in addition to the objects appearing in the video. We show that video clusters found by the cooperation between a human and the machine are better than those obtained through our previous fully automated approach.

Finally, we provide a method for navigating within a group of videos that are looking at the same place. From a current video, the proposed approach allows users to trace a requested trajectory and redirects them to another video in which this trajectory is longer, more detailed, and therefore more observable. To this end, we learn correspondence maps between regions of each video, and use them to reformulate the trajectory drawn by the user into its equivalent in the other views.

In order to validate our approaches, we identify several data sets that are relevant in our context. However, the number of cameras with overlapping fields of view is generally limited. We therefore propose a multi-view data set that we have annotated and which contains 25 cameras located on a university campus, 19 of which have overlapping fields of view. The results obtained through our three contributions are encouraging.

Remerciements

Commencer une thèse peut être dépaysant. Une nouvelle ville, une nouvelle équipe de travail, une nouvelle façon de travailler, autant de facteurs aussi excitants qu'effrayants. Un véritable plongeon dans l'inconnu le plus total. Du moins, j'imagine. Dans mon cas, ce dépaysement consistait à revenir au laboratoire de recherche où j'ai effectué mes stages de deuxième et troisième années d'école d'ingénieur, au sein d'une équipe composée pour la plupart de mes enseignants de l'époque et d'anciens camarades de promo, eux aussi devenus doctorants. Il ne reste plus grand chose du plongeon ou de l'inconnu le plus total.

En premier lieu, je tiens à remercier tout particulièrement mon équipe d'encadrement. Un premier grand merci à toi Vincent, pour tes flots d'idées à chaque brainstorming, qui, après filtrage des plus folles, s'avéraient très pertinentes. Pour tes encouragements et tes remarques toujours positives. Pour nous avoir toujours incités à publier et à voyager sans jamais que le budget de l'équipe ne soit un frein. Un grand merci également à toi Alain pour tes anecdotes inépuisables qui rythmaient nos réunions, racontées avec passion. Pour avoir toujours réussi à concilier tes responsabilités de directeur de département à l'UPS et nos réunions hebdomadaires, souvent à l'ENSEEIHT. Pour tes listes d'articles à lire à n'en plus finir à chaque nouvelle problématique explorée. Enfin, pour tes relectures minutieuses auxquelles aucune coquille ne survit. Merci à toi, Sylvie. Pour ton encadrement pendant le stage de deuxième année d'école d'ingénieur, avec Géraldine et Sandrine. Pour m'avoir offert cette opportunité en me proposant cette thèse et pour m'avoir fait confiance en me sélectionnant parmi les candidats. Pour avoir toujours été de bon conseil et réactive même lorsque le temps te manquait. Pour m'avoir permis de m'impliquer dans certains de tes enseignements et pour ton enthousiasme vis-à-vis de mes idées de sujets, même sur des domaines qui te sont peu familiers. En une phrase : pour avoir été une amie bienveillante tout au long de la thèse et après.

Je tiens aussi à remercier Laure Tougne et Michèle Gouiffès pour avoir accepté de rapporter sur mes travaux de thèse. Leurs retours m'ont été d'une aide précieuse pour préparer la soutenance et envisager des pistes d'amélioration. Je remercie également William Puech pour avoir accepté d'être examinateur et président de jury lors de ma soutenance, ainsi que pour son mail d'encouragement la veille.

Merci à mes collègues de l'IRIT du projet VICTORIA, Florence, André, Christine, Julien et tout particulièrement Patrice et Geoffrey, qui ont partagé avec moi la galère des hackathons à Bochum. Nos réunions ont toujours été intéressantes et fructueuses, dans une ambiance à la fois sérieuse et décontractée. Ce fut un plaisir de publier avec vous sur des thèmes en périphéries de mon sujet de thèse, me permettant de prendre du recul sur mes travaux. Un merci tout particulier à toi, Florence. D'une part, pour

avoir accepté d'intervenir en tant qu'invitée à ma soutenance de thèse. D'autre part, pour m'avoir proposé de participer à l'atelier de médiation en première année de thèse. La soirée espace game avec tous les participants était vraiment sympa.

Je remercie mes collègues doctorants et permanents pour leur bonne humeur et leur bienveillance quotidiennes, Arthur, Bastien, Damien, Jean, Julien, Matpiz, MatLB, Vincent, Marie, Rémy, Axel, Jean-Denis, Pierre, Simone, Géraldine, Nicolas et Sandrine. Je remercie particulièrement Sonia et Thibault, mes collègues de bureau, pour avoir supporté ma présence pendant trois ans. Un grand merci à toi, Thomas-san, collègue, débuggeur à temps plein et avant tout ami depuis ce stage au Japon où l'on a rencontré des gens formidables, des familles d'accueil adorables, où un inconnu nous a offert une pastèque, et tant d'autres histoires qu'on ressasse sans cesse sans se lasser. Un merci également au groupe de langue des signes du jeudi midi ainsi qu'à nos secrétaires qui font un travail incroyable : Sylvie, Muriel et Anabelle.

Un grand merci à mes meilleurs amis, rencontrés sur des jeux en ligne. Pour ces années de rigolade ensemble. Parce qu'ils ont répondu présent sans hésiter quand je leur ai proposé qu'on voyage au Japon ensemble. C'était les meilleures vacances de ma vie. Merci à vous Divin, Flora, Letana et Pablo.

Merci également à ma famille pour leur soutien, à ma mère qui m'a toujours laissé faire mes choix, à ma tante et ses délicieux repas de Noël chaque année, mes frères et mes cousins, ainsi qu'à Sylvain, désormais officiellement mon beau-père.

Enfin, merci à la personne qui m'est la plus chère et que j'aime plus que tout, Arwen, pour avoir illuminé ma vie.

Table des matières

Liste des figures

Liste des algorithmes

Liste des tableaux

Chapitre 1

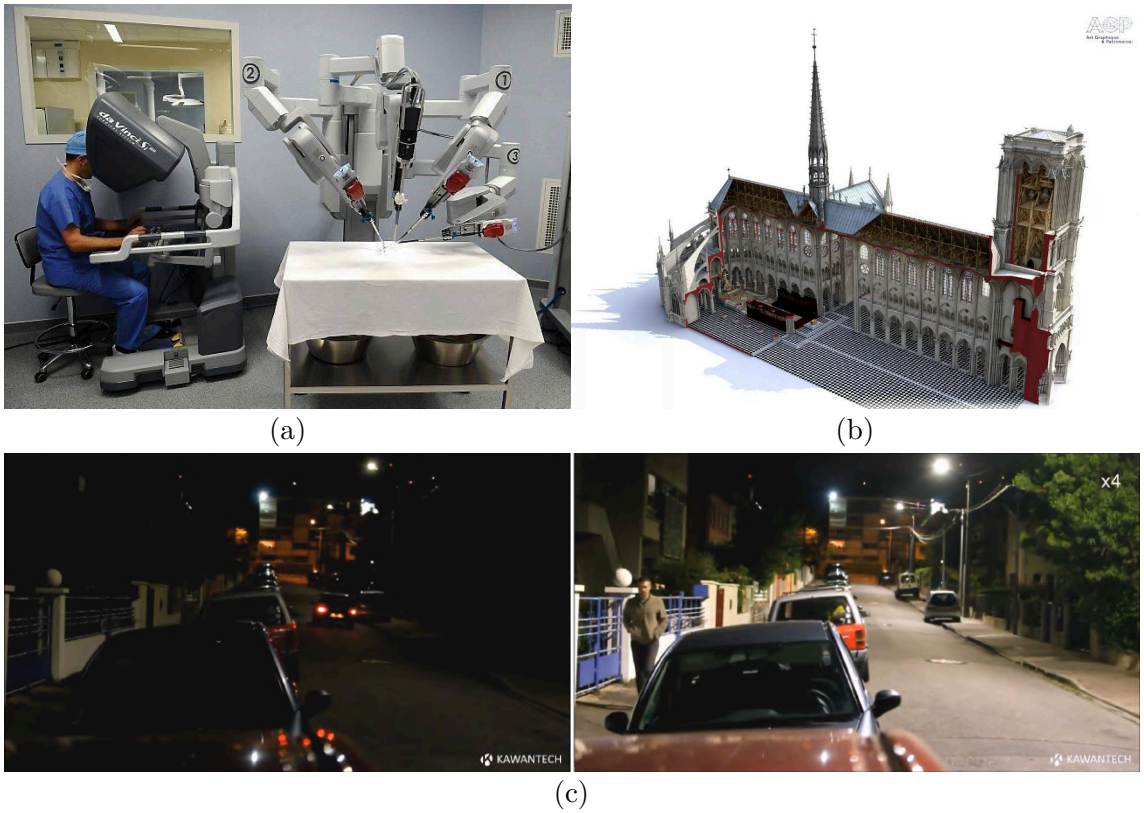
Introduction

Contenu

1.1 Préambule

Au cours des vingt dernières années, les progrès de l'informatique ont bouleversé nos sociétés. Les capacités de stockage et de calcul ont été décuplées tandis que le coût des machines a chuté, devenant de plus en plus accessibles au grand public. Les contenus multimédia (musiques, images, vidéos) sont progressivement devenus plus faciles à produire et à partager, au point qu'en 2020, ce sont 720 000 heures de vidéos qui sont ajoutées quotidiennement sur la plateforme de partage de vidéos `Youtube` : visionner l'ensemble des vidéos publiées en une seule journée demanderait d'y consacrer une vie entière à plein temps ! Le nombre de vidéos proposées comme résultats de recherche de certains évènements peut atteindre la centaine, qu'il s'agisse d'un concert, d'une convention ou d'une manifestation. Cette surabondance de contenus permet d'avoir à disposition une plus grande diversité d'informations visibles à l'écran, ainsi qu'une plus grande variété de points de vue de la scène. Mais trouver une information particulière dans un tel volume de données peut se révéler très chronophage. D'un domaine à un autre, pouvoir déceler rapidement l'information pertinente peut avoir plus ou moins d'impact. Dans le domaine sportif, les matchs sont généralement filmés simultanément par de nombreuses caméras : l'enjeu est alors de diffuser le point de vue le plus pertinent à tout instant. Par exemple, dans un match de football, on peut alterner entre des plans larges lorsque le ballon se situe vers le centre du terrain et des plans plus serrés lorsqu'il est proche des cages. Le but est alors de garder l'attention des spectateurs et l'impact est avant tout économique. Dans le domaine de la défense ou de la vidéoprotection, il est crucial de pouvoir analyser rapidement un ensemble de vidéos pour en isoler certaines informations essentielles : ce sont parfois des vies qui en dépendent. Là encore, les récents progrès de l'informatique ont permis d'importantes avancées dans l'analyse d'images et de vidéos, et plus généralement en vision par ordinateur.

Depuis une vingtaine d'années, le domaine de la vision par ordinateur connaît un essor important. En effet, le nombre de publications scientifiques soumises chaque année à la conférence *Computer Vision and Pattern Recognition*, l'une des plus prestigieuses du domaine, a été multiplié par 10 (un peu plus de 500 en 2000 contre 5 160 en 2019). Les avancées dans ce domaine ont permis à des chirurgiens d'opérer des patients à distance à l'aide de dispositifs de réalité augmentée reproduisant les tissus et les organes de la personne opérée. Elles permettent également à des historiens de contribuer à la sauvegarde du patrimoine en numérisant des objets ou des bâtiments sous forme de modèles 3D. Enfin, elles permettent à certaines villes de réduire leurs dépenses d'éclairage public la nuit en équipant les lampadaires de caméras capables de détecter la présence ou non d'une personne ou d'un véhicule, et en adaptant l'intensité lumineuse en fonction. Ces quelques exemples sont illustrés par la figure 1.1 et montrent, par leur grande diversité, l'étendue de ce que permet la vision par ordinateur, la rendant d'autant plus difficile à définir.



(a) (b)

(c)

FIGURE 1.1 – Quelques applications de la vision par ordinateur – (a) Simulation d'une intervention chirurgicale à distance : le chirurgien s'installe à la console et manie à distance les bras d'un robot (photographie de Franck DUBRAY pour Ouest France), (b) une modélisation 3D de la cathédrale de Notre-Dame de Paris (source : Arts graphiques et patrimoine), (c) le dispositif `Kara` de la société `Kawantech` permet aux lampadaires de n'éclairer qu'en présence de piétons. À gauche, un véhicule circule et les lampadaires sont éteints. À droite, un piéton marche et les lampadaires sont allumés. Les deux images sont des captures d'écran issues d'une vidéo de la chaîne `Youtube Kawantech`.

La vision par ordinateur consiste à reproduire notre système de perception visuelle en couplant un système d'acquisition d'images ou de vidéos avec un ordinateur ou, de manière plus intuitive, à apprendre à un ordinateur à « voir » comme un humain, à comprendre la scène contenue dans une image. Après des années d'entraînement, notre cerveau est capable d'interpréter des images, d'y reconnaître des objets et des personnes, de décrire leurs actions, ou encore de se représenter la façon dont sont positionnés les éléments les uns par rapport aux autres. Pour la machine, ces mêmes images ne sont que des suites de valeurs numériques représentant la couleur de chaque pixel. Ainsi, de façon très générale, le but de la vision par ordinateur est de donner du sens à des suites de nombres qui encodent des images au travers de traitements informatiques. De manière plus détaillée, les principaux objectifs de la vision par ordinateur sont les suivants :

— la détection d'objets, qui consiste à indiquer la présence d'un objet dans une image

et à le localiser au sein de l'image ;

— le suivi d'objets, qui consiste à retrouver les différentes instances d'un même objet dans le temps au sein d'une même vidéo ;

— la reconnaissance d'actions, qui consiste à déterminer l'action réalisée par une personne sur une séquence vidéo ;

— la réidentification, qui consiste à reconnaître un objet apparaissant dans différentes vidéos ;

— la perception du relief, qui consiste à estimer la structure de la scène en trois dimensions à partir d'images présentant des projections de cette scène en deux dimensions.

Selon l'application visée, il peut être nécessaire de combiner plusieurs de ces tâches pour permettre à un ordinateur d'interpréter finement un ensemble de vidéos. Illustrons cela avec un exemple d'application en particulier : une enquête.

L'analyse des contenus de vidéos collectés est devenue un élément essentiel lors d'enquêtes liées à un acte criminel ou à une attaque terroriste. La quantité de données vidéo d'intérêt disponibles lors d'enquêtes légales ne cesse de croître avec le déploiement des équipements vidéo, des caméras de surveillance dans des zones publiques et privées, des caméras portées par les forces de police, des smartphones et des caméras numériques personnelles. Cette analyse devient donc de plus en plus délicate du fait de la quantité d'informations à traiter et de la variabilité des sources vidéo. Dans un contexte d'enquête, les premières heures sont décisives, or la visualisation exhaustive de toutes ces vidéos par des opérateurs humains est une tâche longue et coûteuse. Des solutions sont donc nécessaires pour permettre à des enquêteurs de visionner plus efficacement une collection de vidéos, en automatisant certaines tâches. Concrètement, dans ce cadre, la vision par ordinateur peut apporter les possibilités suivantes :

— Elle peut permettre d'isoler des moments intéressants dans les vidéos : sur les milliers d'heures de vidéos de caméras de surveillance, les moments d'intérêt, où il y a des personnes visibles, ne représentent qu'une petite partie. Dans ce cas, les tâches impliquées sont la détection d'objets et de mouvement.

— Elle peut permettre de suivre le mouvement d'une personne ou d'un véhicule dans une vidéo, par exemple si l'on veut détecter qu'un individu est entré dans un bâtiment par une porte de sortie, ou pour détecter le passage d'un véhicule par un sens interdit. La seule détection des objets risquerait de donner de nombreuses fausses alarmes à cause des personnes et des véhicules empruntant le chemin dans le sens habituel, aussi est-il nécessaire d'analyser le mouvement des objets. Dans ce cas, il s'agit d'une tâche de suivi.

— À partir de l'apparence d'une personne ou d'un objet aperçu dans une vidéo, par exemple un individu suspect ou une arme, elle peut aider à retrouver cette personne

ou cet objet dans une autre vidéo. Il s'agit alors d'une tâche de réidentification.

— Enfin, sous certaines conditions, elle peut fournir une reconstruction 3D de la scène qui essaye de reproduire l'environnement réel avec autant de fidélité que possible en combinant les informations disponibles dans plusieurs vidéos observant la scène depuis des points de vue différents. Il s'agit là d'une tâche de détermination du relief.

Ce dernier exemple met en évidence l'apport que peut fournir la vision par ordinateur lorsque l'on dispose de grandes quantités de données vidéo à traiter et en particulier dans un contexte d'enquête. En pratique, pourtant, les méthodes employées n'offrent pas toujours des résultats exploitables. L'émergence de projets de recherche visant à faciliter l'analyse de grandes quantités de vidéos montre qu'il reste encore de la place pour l'innovation dans ce domaine. Le projet européen VICTORIA, dans lequel s'inscrivent les travaux présentés dans ce mémoire, est l'un d'entre eux.

1.2 Projet VICTORIA

Les travaux présentés dans ce mémoire ont été financés par le programme de recherche et d'innovation Horizon 2020 de l'Union européenne dans le cadre du projet VICTORIA, au titre de la convention de subvention n°740 754. Le projet VICTORIA (***V**ideo Analysis For **I**nvestigation of **C**riminal and **T**err**ori**st **A**ctivities*) vise à réaliser une avancée dans l'examen de grands nombres de vidéos relatant un même évènement. Il a débuté en mai 2017 pour une durée de trois ans.

Le constat de départ était le suivant :

— Il n'existe pas d'outil d'investigation qui soit largement utilisé par les différentes polices des états européens : chaque pays utilise ses propres outils. Cela peut poser des problèmes de coordination lors d'enquêtes impliquant plusieurs pays.

— Les outils existants sont développés et vendus par des sociétés privées. Outre le coût important de tels outils, le code source n'est pas rendu accessible. La maintenance et l'ajout de nouvelles fonctionnalités dépend alors entièrement d'un acteur

indépendant.

— Les fonctionnalités disponibles en termes d'analyse vidéo sont assez limitées et sont peu robustes aux mouvements de caméras, aux différences d'angles de vue, ainsi qu'aux différences de qualité d'image.

— Très peu de données représentatives, telles que des reproductions de scènes de crime ou des vidéos réelles de caméras de surveillance, sont disponibles. Or, elles sont nécessaires pour faire avancer la recherche et le développement.

Ainsi, il semble important de développer des outils d'investigation libres qui soient communs aux différentes polices d'Europe, qui offrent des fonctionnalités d'analyse de vidéos, efficaces et robustes aux différences de caractéristiques entre vidéos, pour les aider dans leurs tâches quotidiennes.

Quatorze partenaires européens ont été impliqués dans le projet autour de dix lots (*work packages*) composés de tâches auxquelles sont associées un ou plusieurs partenaires. Le détail des lots est donné dans l'annexe A. Les travaux de cette thèse ont été menés en lien avec le lot qui concerne l'analyse vidéo. L'objectif de ce lot est de fournir des outils qui facilitent la visualisation de la scène et la navigation au sein de celle-ci. Initialement, les enquêteurs disposent d'un ensemble de vidéos dont ils veulent extraire des informations utiles à leur enquête. Partant d'une vidéo de départ qui présente de l'intérêt, ils peuvent vouloir trouver d'autres vidéos qui observent la même scène depuis un autre point de vue et montrent potentiellement des indices qui n'étaient pas visibles sur la vidéo de départ. En l'absence d'outil d'aide à l'investigation, ils n'ont pas d'autre choix que de visionner l'intégralité des vidéos de manière exhaustive pour trouver celles qui observent la même zone. La solution que propose le projet VICTORIA consiste à effectuer, lorsque cela est possible, une reconstruction de la scène composée, d'une part, des parties statiques de la scène (le décor) et, d'autre part, des parties dynamiques, c'est-à-dire en mouvement au cours du temps (les objets). La reconstruction étant en trois dimensions enrichie du temps pour les parties dynamiques, elle est qualifiée de reconstruction 4D. Elle offre une synthèse de la scène qui capture un maximum des éléments visibles dans les différentes vidéos, tout en proposant une visualisation intuitive, dans la mesure où elle tend à reproduire l'environnement réel aussi fidèlement que possible. Ainsi, au lieu de regarder une à une les vidéos de façon séquentielle, l'utilisateur peut avoir une vue globale réunissant plusieurs vidéos à la fois. De plus, l'utilisateur peut naviguer à travers la scène reconstruite au moyen d'une caméra virtuelle et se déplacer en avant ou en arrière dans le temps, guidé par des indices visuels et par sa compréhension progressive de ce qui s'est passé avant, pendant, et après le crime. Dans ce cas idéal, les enquêteurs disposent alors d'un point de vue omniscient de la scène.

Nos contributions directes au projet concernent la reconstruction des parties dynamiques. Il s'agit de plusieurs documents livrables et d'une partie conséquente de déve-

loppement d'un module d'estimation de squelettes 3D de personnes. Nous ne détaillerons pas davantage ces travaux dans le corps de ce mémoire car ils ne présentent pas de dimension recherche. Nous apporterons des précisions en conclusion dans la section 6.1.1, lors d'un bilan de toutes nos contributions.

1.3 Problématique

En pratique, il n'est pas toujours possible d'effectuer la reconstruction d'une scène. À titre d'exemple, nous avons utilisé le logiciel de reconstruction 3D `Meshroom` qui utilise la librairie `AliceVision`[1] pour tenter de reconstruire une scène filmée par 17 caméras dont les vues sont présentées sur la figure 1.2.

FIGURE 1.2 – 17 vues du jeu de données ToCaDa [Malon 18] – Les caméras observent toutes la même scène et certains éléments saillants, tels que les places de parking en bleu, sont visibles et facilement distinguables dans toutes les vues.

En raison de la trop grande diversité de points de vue des caméras, les algorithmes utilisés, dits de *Structure-from-Motion*, ont été incapables de fournir une reconstruction 3D de la scène. Nous sommes parvenus à obtenir une reconstruction 3D d'une partie de la scène et de la façade du bâtiment en enrichissant les points de vue d'une vingtaine de photos, visibles sur la figure 1.3 prises successivement à quelques mètres d'intervalle et en orientant l'appareil photo en direction du bâtiment. Cette reconstruction est présentée sur la figure 1.4.

Ces conditions restrictives motivent la nécessité de disposer de méthodes alternatives aidant à établir les liens entre différentes vidéos d'un ensemble, à naviguer entre elles et à en visualiser le contenu. C'est le sujet que nous nous proposons de traiter dans cette thèse.

De manière très générale, dire qu'il existe un lien entre deux vidéos peut avoir une multitude de significations différentes. Des vidéos filmées par une même personne, à une même date ou encore en un même lieu sont liées chacunes à leur façon, respectivement par la personne, la date ou le lieu. En fait, la notion de lien entre vidéos dépend de la

1. https://alicevision.org/

FIGURE 1.3 – Photos supplémentaires — En plus des vues de la figure 1.2, nous avons pris une vingtaine de photos de la façade du bâtiment à des positions et selon des orientations proches.

personne qui recherche ces liens et du contexte d'application. Nous nous plaçons dans le contexte d'enquête décrit plus haut. Afin de définir la notion de lien entre vidéos dans ce contexte, nous pouvons nous demander ce que serait une paire de vidéos non liées. Dans un travail d'invesgation, une telle paire est caractérisée par le fait que depuis l'une des vidéos, des opérateurs n'ont aucun intérêt à accéder à l'autre vidéo de la paire. Cela signifie qu'aucun élément n'apparaît en commun entre les deux vidéos et qu'elles observent probablement des scènes différentes. À l'inverse, un lien entre deux vidéos signifie donc qu'elles présentent des éléments en commun et qu'il est donc pertinent de regarder le contenu de l'autre vidéo pour obtenir potentiellement plus d'informations si le contenu de la première présente de l'intérêt. Les liens dépendent donc uniquement du contenu des vidéos : on n'exploitera pas les métadonnées des fichiers telles que les coordonnées géographiques dont on ne dispose pas dans le cas général. Lorsque deux vidéos présentent effectivement des entités en commun, et en supposant qu'elles sont synchronisées sur une horloge commune, cela peut signifier :

— ou bien que la région dans laquelle apparaît l'entité est commune aux champs de vue des deux caméras si les entités sont présentes simultanément au même instant dans les deux vidéos ;

— ou bien que les caméras sont spatialement proches mais que leurs champs de vue sont disjoints s'il y a un délai temporel entre l'apparition de l'entité dans une caméra et son apparition dans l'autre caméra.

Les travaux présentés dans cette thèse traiterons essentiellement le cas des liens de recouvrement des champs de vue, que nous qualifierons plus simplement de **liens de recouvrement**. Automatiser la recherche d'éléments communs est une tâche difficile en raison de la différence des points de vue des caméras, de la différence d'éclairage entre les vues et de potentielles occultations : une personne qui apparaît dans une vue peut avoir une ap-

FIGURE 1.4 – Reconstruction 3D de la façade de l'IRIT obtenue en ajoutant aux vues présentées sur la figure 1.2 la vingtaine de photos de la figure 1.3.

parence assez différente dans une autre. Des techniques sont donc nécessaires pour décrire et comparer les contenus de deux vidéos afin de déterminer si elles présentent ou non un lien de recouvrement.

L'automatisation de la recherche de liens a pour but d'accélérer le travail des opérateurs. Cependant, pour cela, le résultat fourni par l'algorithme se doit d'être fiable. En cas d'erreur, c'est-à-dire en cas de lien détecté erroné ou de lien manquant, le travail des opérateurs peut être perturbé et la confiance qu'ils font à la machine en pâtit. Pour éviter une telle situation, il est important que l'algorithme travaille de concert avec les opérateurs, supposés experts du domaine, en les sollicitant lors de la recherche automatisée de liens sur des cas difficiles et en apprenant progressivement de leur expertise. L'objectif de trouver les liens entre vidéos est de pouvoir ensuite s'en servir pour naviguer d'une vidéo à une autre. Depuis une vidéo courante affichée à l'écran, il faudra envisager les types de requêtes que pourraient formuler des opérateurs, notamment en termes de trajectoires pour suivre le déplacement d'objets entre caméras.

Si nous récapitulons, la problématique est donc la suivante : lorsqu'il n'est pas possible d'obtenir de reconstruction de la scène, comment faciliter la compréhension de la scène, la recherche d'éléments et la navigation entre vidéos ? Quels traitements permettent de trouver des liens entre vidéos ? Comment décrire et comparer le contenu de différentes vidéos ? Faire coopérer les opérateurs avec la machine permet-il d'obtenir de meilleurs résultats sans nécessiter trop de travail ? Quels types d'interactions est-il pertinent de proposer à l'utilisateur pour faciliter son travail ?

1.4 Organisation du mémoire

Ainsi, l'organisation de ce mémoire suit ce questionnement et s'articule autour de cinq autres chapitres.

Tout d'abord, dans le chapitre 2, nous présentons les différents types de liens qui existent et les approches qui ont été proposées pour les déterminer. Nous abordons ensuite plus en détails les liens entre caméras dont les champs de vue se recouvrent et décrivons une première contribution visant à caractériser et comparer deux vidéos en fonction de leurs contenus pour déterminer si elles sont liées ou non. Nous nous servons pour cela de détecteurs d'objets et présentons différentes approches de détection et de réidentification.

Les expérimentations conduites sur cette première contribution sont présentées dans le chapitre 3. Nous commençons par passer en revue les jeux de données vidéos multicaméras existants et comparons leurs propriétés pour ne conserver que ceux qui sont exploitables pour notre cas d'étude. Nous présentons également un jeu de données que nous avons créé et annoté et qui présente 19 caméras dont les champs de vue s'intersectent et 6 autres caméras aux champs de vue disjoints. Enfin, nous présentons le protocole d'évaluation et les résultats de notre approche d'établissement des liens de recouvrement entre champs de vue des caméras.

Le chapitre 4 vient compléter cette approche et présente une deuxième contribution. Nous mettons en place une coopération entre l'homme et la machine par un mécanisme d'apprentissage actif en sollicitant l'utilisateur pour des cas difficiles et en lui permettant de corriger d'éventuelles erreurs de la machine. Nous exploitons également l'arrière-plan des vidéos en plus des objets qui y apparaissent. Nous décrivons un protype d'interface qui permet à l'utilisateur de voir les groupements de vidéos se former progressivement et au travers de laquelle il peut interagir et répondre aux sollicitations de la machine. Nous présentons les résultats obtenus grâce à cet apprentissage actif des liens et les comparons avec les résultats d'algorithmes non supervisés.

Dans le chapitre 5, nous proposons une troisième contribution. Il s'agit d'un mécanisme de navigation au sein d'un groupe de vidéos dont les champs de vue se recouvrent permettant de passer de l'une à l'autre par une requête trajectographique. Nous proposons pour cela une interface dans laquelle l'utilisateur peut, depuis une vidéo courante, tracer une trajectoire requête et être redirigé vers une autre vidéo dans laquelle cette trajectoire est plus longue, plus détaillée et donc mieux observable. Nous apprenons pour cela des cartes de correspondances entre régions de chaque vidéo et les utilisons pour reformuler la trajectoire tracée par l'utilisateur en son équivalent dans les autres vues. Nous présentons les résultats obtenus en termes de reformulation de trajectoire et de navigation entre vidéos.

Enfin, nous concluons ce mémoire dans le chapitre 6 en mettant en avant l'intérêt des méthodes que nous avons proposées et nous présentons les perspectives de recherche envisagées pour compléter et enrichir nos premières contributions dans le domaine.

Chapitre 2

Recherche de liens entre vidéos

Contenu

2.1 Introduction

Nous l'avons précisé dans le chapitre 1, la notion de « lien » entre vidéos est très générale, ce qui la rend difficile à définir. Dans le contexte qui est le nôtre, nous souhaitons que les liens entre vidéos aident à accélérer le travail d'enquêteurs en les guidant dans le visionnage des vidéos. Depuis une vidéo qu'ils estiment d'intérêt, c'est-à-dire dont le contenu apporte des informations utiles à l'enquête, nous souhaitons leur recommander d'autres vidéos qui apportent des informations complémentaires : c'est en ce sens que nous parlerons de vidéos **liées**. Plus précisément, un lien entre deux vidéos signifie que des personnes ou des objets qui apparaissent dans l'une des vidéos ont tendance à apparaître aussi dans l'autre, soit simultanément, soit après un certain temps. Ainsi, si des enquêteurs sont intéressés par des éléments dans une vidéo, il peut être pertinent de leur suggérer de visionner les autres vidéos qui lui sont liées car il est probable que ces éléments d'intérêt y apparaissent également. Ces éléments communs peuvent être de différentes natures : il peut s'agir de points d'intérêt que l'on cherche à apparier au vu de leurs similarités de voisinages ou d'entités concrètes, généralement des personnes ou des véhicules. La recherche de liens entre vidéos consiste alors à déterminer, pour chaque paire de vidéos d'un ensemble, si des éléments présents dans la première vidéo apparaissent également dans la seconde dans un intervalle de temps donné. Le fait de trouver un lien entre deux vidéos apporte un renseignement sur la proximité spatiale des caméras ayant filmé ces vidéos et donc sur la topologie du réseau de caméras. Cette proximité spatiale est directement liée à l'intervalle de temps considéré lors de la recherche d'éléments communs entre vidéos.

D'autre part, dresser la liste des liens peut aider à suivre le déplacement au sein du réseau de caméras d'une entité aperçue dans l'une des vidéos en prédisant dans quelles autres vidéos elle est susceptible d'apparaître et dans quelles vidéos elle a peu de chances d'apparaître, soit simultanément, soit après un certain temps qui peut varier de quelques secondes (le long d'un couloir par exemple) à quelques heures (pour des caméras positionnées régulièrement sur une autoroute). Tout au long de cette thèse, nous ferons l'hypothèse que toutes les vidéos sont synchronisées temporellement, c'est-à-dire que l'on peut dater chaque image de chaque vidéo selon une horloge commune. Pour une paire de caméras donnée, il existe trois possibilités, illustrées par la figure 2.1 :

1. Les champs de vue des caméras peuvent se croiser (caméras 1 et 2) — il existe alors des régions communes aux deux vidéos et un objet passant dans l'intersection des champs de vue est visible simultanément dans les deux vidéos à la fois.
2. Les champs de vue des caméras ne se croisent pas mais les caméras sont spatialement proches à une échelle donnée (caméras 3, 4 et 5). À l'échelle d'un bâtiment, on peut imaginer des caméras placées dans chaque pièce et chaque couloir : les champs de vue des différentes caméras ne s'intersectent pas mais certains objets apparaissent dans différentes caméras avec un délai temporel de transit qui dépend de la nature

et des caractéristiques des objets considérés.

3. Les champs de vue des caméras ne se croisent pas et les caméras ne sont pas spatialement proches. Aucun objet n'apparaît en commun dans les deux vidéos dans un intervalle de temps donné. Elles sont alors totalement disjointes. C'est le cas par exemple des caméras 2 et 3.

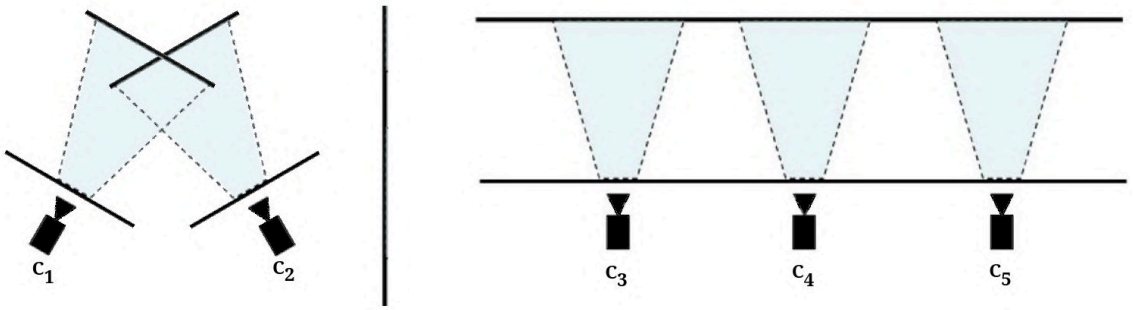


FIGURE 2.1 – Relations possibles entre caméras – Les champs de vue des caméras 1 et 2 se croisent, les objets apparaissant dans la région commune sont visibles simultanément dans les deux caméras. Les champs de vue des caméras 3, 4 et 5 ne se croisent pas, mais leur proximité spatiale fait qu'un objet aperçu dans l'une des caméras a une forte probabilité de passer dans le champ de vue des autres caméras après un certain délai. Enfin, la barre verticale indique que les caméras 1 et 2 sont disjointes des caméras 3, 4 et 5.

Les travaux présentés dans ce mémoire concernent majoritairement le cas des ensembles de vidéos avec recouvrement. Il semble toutefois pertinent de dresser l'état de l'art des approches proposées pour traiter le cas sans recouvrement pour s'inspirer des techniques proposées. Nous décrirons ensuite l'état de l'art des approches concernant le cas avec recouvrement. Dans les deux cas, les approches s'appuient sur le contenu des vidéos, notamment sur les objets qui y apparaissent. Nous présentons dans un premier temps les approches de détection et de réidentification les plus liées à notre travail.

Concernant la terminologie, nous emploierons le terme **vidéo** pour désigner une séquence d'images. Le terme **caméra** désignera quant à lui un dispositif d'acquisition vidéo et sera employé pour parler de l'objet physique.

2.2 Éléments permettant de décrire le contenu d'une vidéo

La plupart des approches de recherche de liens entre caméras exploitent deux aspects complémentaires des vidéos :

— l'arrière-plan, c'est-à-dire les parties statiques de la scène qui correspondent au décor ;

— les objets, généralement animés.

Ces deux notions ne sont pas toujours faciles à définir et donc pas faciles à distinguer. À titre d'exemples, un véhicule stationné pendant toute la durée d'une vidéo est-il considéré comme un objet ou fait-il partie du décor ? La même question peut se poser pour un brin d'herbe s'agitant au gré du vent. Pour répondre à ces questions, repartons de ce que nous cherchons à faire : nous voulons trouver des liens entre vidéos à partir de la similarité de leurs contenus. Ces contenus doivent alors être suffisamment caractéristiques et facilement discernables. Le mouvement est également intéressant : de par notre hypothèse d'horloge commune aux différentes caméras, une entité en mouvement dans une vidéo le sera simultanément dans toutes les autres vidéos qui la contiennent. Toutefois, l'amplitude du mouvement et la taille de l'objet doivent être suffisamment grandes pour permettre de caractériser l'objet de manière significative, c'est-à-dire de manière suffisamment détaillée pour que cela soit discriminant pour le reconnaître dans les différentes vidéos. Ainsi, bien qu'en mouvement, le brin d'herbe est peu discriminant et de trop petite taille pour être considéré comme un objet à part entière. De son côté, la voiture stationnée est un objet facilement discernable. Cependant, du fait de son immobilitéé et à défaut de pouvoir déterminer l'apparence de la partie du décor qu'elle cache, elle peut être considérée comme faisant partie du décor.

2.3 Détection et réidentification des objets

Plusieurs des approches de recherche de liens entre vidéos s'appuient sur la détection et la reconnaissance ou la réidentification d'objets apparaissant dans les vidéos. La détection consiste à localiser une ou plusieurs occurrences d'objets d'une certaine catégorie dans une image. À partir d'un objet détecté dans une image, la réidentification consiste à trouver les occurrences de cet objet particulier dans d'autres images. La bibliographie sur ces deux sujets est vaste et nous proposons une présentation succincte des éléments principaux les plus liés à notre travail. De manière très générale et simplifiée, nous pouvons dire que les méthodes de détection caractérisent des régions d'une image par un vecteur, puis comparent ce vecteur à des modèles de catégorie d'objet appris grâce à de nombreuses images de ce type d'objet. La réidentification, quant à elle, compare le vecteur caractéristique d'un objet donné issu d'une image aux vecteurs caractéristiques des objets détectés dans les autres vidéos et les classe du plus similaire, c'est-à-dire celui qui a la plus forte probabilité de correspondre au même objet, au moins similaire. Dans les deux types de méthode, les deux grandes étapes sont donc :

1. le calcul d'un descripteur ;
2. la comparaison du descripteur à d'autres descripteurs ou à un modèle.

Cette thèse ne présentant pas directement de contribution sur les aspects de détection

et de réidentification, nous donnons une présentation synthétique des approches les plus utilisées. Toutefois, plus de détails sont fournis dans l'annexe B sur les critères d'évaluation des approches.

2.3.1 Introduction sur les descripteurs

D'un point de vue informatique, une image est représentée par un tableau de chiffres encodant le niveau de gris (ou la couleur) de chaque pixel. Une telle représentation est difficile à manipuler telle quelle pour détecter ou réidentifier des objets. Pour mesurer la similarité entre deux images de même taille I_1 et I_2 directement à partir de leurs niveaux de gris, il est par exemple possible de calculer la somme des différences absolues :

$$\sum_i \sum_j |I_1(i,j) - I_2(i,j)|. \tag{2.1}$$

Ce score est d'autant plus proche de 0 que les niveaux de gris des pixels sont deux à deux proches. Cette approche, très simple, nécessite que les images soient très similaires, de même taille et qu'il n'y ait pas de décalage géométrique entre elles. En effet, même un léger décalage peut entraîner une forte augmentation du score. Pour faire face à cette faiblesse, il est possible de considérer des méthodes par fenêtre glissante où l'on fait avancer une image sur l'autre en toutes les positions possibles et où l'on calcule à chaque fois un score de comparaison des images pour obtenir une carte des scores pour tous les décalages possibles. L'approche devient ainsi robuste au décalage et peut être utilisée sur des paires d'images de tailles différentes. En revanche, il est toujours nécessaire que les images des mêmes objets soient très similaires. En pratique, c'est rarement le cas : un même objet vu depuis deux points de vue différents peut avoir des apparences très différentes dans les deux images. Il peut être visible entièrement dans une vue et occulté dans l'autre, ou être bien éclairé dans une vue et moins dans l'autre. Il est alors possible d'utiliser des mesures de similarité robustes aux changements de luminosité et robustes aux occultations [Chambon 11]. Toutefois, ces mesures ne permettent pas de traiter les changements d'apparence. Pour traiter correctement ces cas difficiles, d'autres propriétés de l'image, telles que la texture ou les contours, peuvent être exploitées. En vision par ordinateur, le terme de **descripteur** est utilisé pour qualifier ces différentes propriétés extraites. Le rôle d'un descripteur est de capturer certaines propriétés d'une partie d'une image (on parle alors de descripteur local) ou d'une image entière (on parle alors de descripteur global). Ces propriétés sont généralement stockées dans un vecteur qui présente une structure précise : chaque valeur correspond à un type de propriété particulier. Cette représentation des propriétés des images sous forme de vecteurs de $\mathbb{R}^n$ permet d'obtenir une représentation de taille constante. Nous allons maintenant décrire plusieurs familles de descripteurs utilisés pour la détection et la réidentification en présentant à chaque fois un exemple

de descripteur. Nous présenterons ensuite différentes façons dont ces descripteurs peuvent être utilisés pour détecter une catégorie d'objet ou réidentifier un objet.

2.3.2 Descripteurs par masques

Il s'agit d'une des techniques les plus anciennes et les plus populaires de par sa simplicité d'utilisation. Dans le contexte de ce travail, nous pouvons citer [Papageorgiou 98] pour la détection de visages et de piétons, qui a ensuite été repris et amélioré [Viola 01, Viola 04]. L'approche repose sur le fait qu'en général, les contours d'un objet dans une image se distinguent par un changement du niveau de gris des pixels. Les auteurs considèrent différents masques, illustrés sur la figure 2.2, qu'ils font glisser sur l'image à toutes les positions possibles et toutes les tailles possibles de masques. Pour chaque position et taille d'un masque, ils calculent sa réponse en sommant les valeurs des niveaux de gris des pixels situés dans les zones blanches et en soustrayant les valeurs des pixels situés dans les zones noires. L'ensemble de ces réponses forme un vecteur qui peut être vu comme un descripteur caractérisant la similarité de l'image aux différents masques.

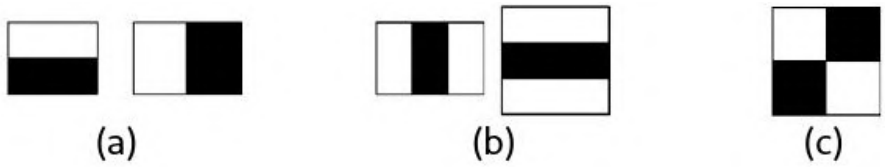


FIGURE 2.2 – Masques utilisés dans [Viola 01] – (a) Masques détecteurs de bords, (b) Masques détecteurs de lignes, (c) Masques détecteurs de lignes en diagonale. Figure extraite de [Viola 01].

2.3.3 Descripteurs de contours

Dans la famille des descripteurs de contours, l'idée est qu'une catégorie d'objet donnée est caractérisable par la forme de ses contours. Proposés dans [Dalal 05], les histogrammes de gradients orientés (HOG) sont utilisés pour décrire la répartition des gradients d'une image. Plus précisément, en suivant le même principe que SIFT, *Scale Invariant Feature Transform* [Lowe 04], le descripteur proposé contient une concaténation des histogrammes obtenus en quantifiant localement les orientations des vecteurs gradients. Ce type de descripteur capture ainsi les contours mais ne contient pas d'information colorimétrique. La figure 2.3 présente notamment un exemple de représentation d'un descripteur HOG.

2.3.4 Modèles rigides par parties

Une des difficultés auxquelles se heurtent de nombreuses méthodes de détection et que nous avons déjà abordées réside dans la grande variabilité des apparences des objets.

En particulier, la détection de personne est particulièrement difficile du fait de l'aspect déformable du corps humain, notamment en raison de la multitude de poses différentes dans lesquelles il peut se trouver. Pour faire face à cette difficulté, les modèles rigides par parties (*part-based modelling*) ont été proposés [Fischler 73, Felzenszwalb 05]. L'idée consiste à représenter un objet comme un ensemble de parties rigides arrangées et liées les unes par rapport aux autres de façon déformable, et à modéliser l'apparence de chaque partie indépendamment. Suite à leur succès, les descripteurs HOG sont rapidement utilisés à cette tâche pour détecter des personnes [Felzenszwalb 08]. Dans cet article, les parties sont au nombre de six et correspondent à la tête, à chaque avant-bras et chaque bras, ainsi qu'aux pieds. Un exemple de modèle en cinq parties basé sur les descripteurs HOG est illustré sur la figure 2.3.

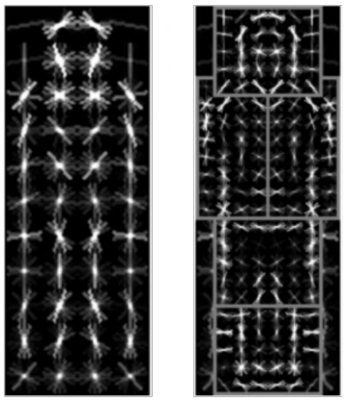


FIGURE 2.3 – Modèles rigides par parties basés sur les descripteurs HOG [Felzenszwalb 05] – À gauche, un modèle global de personne. À droite, un modèle en 5 parties correspondant à la tête, le bras gauche, le bras droit, le haut des jambes et le bas des jambes. Dans cet exemple, les deux types de modèles sont caractérisés par des descripteurs HOG. Les segments blancs représentent la distribution locale des orientations des vecteurs gradients. Image extraite de [Felzenszwalb 09].

2.3.5 Descripteurs colorimétriques

Cette famille de descripteurs exploite le fait que la couleur est une caractéristique importante de la détection ou de la réidentification d'objets. Le descripteur SCNCD (*Salient Color Names based Color Descriptor*) [Yang 14] en est un exemple. Les auteurs définissent 16 couleurs saillantes dans la palette RVB (Rouge, Vert, Bleu) et associent à chaque autre couleur $c = (r, v, b)$ une représentation à partir des 16 couleurs saillantes. Cette représentation est une distribution des couleurs saillantes calculée en fonction de la proximité spatiale entre la couleur et les 5 couleurs saillantes qui lui sont les plus proches dans le cube RVB. À chaque imagette de personne, ils associent alors un descripteur formé de 6 tranches horizontales de même taille qu'ils décrivent par la moyenne des distributions de couleurs saillantes des pixels contenus dans la tranche. Le descripteur final SCNCD est la

concaténation des descripteurs des 6 tranches. L'idée des différentes tranches et des distributions de couleurs saillantes est qu'elles permettent une description proche de celle du langage : lorsqu'on décrit grossièrement une personne, on a tendance à spécifier la couleur des vêtements en distinguant généralement le haut et le bas, et éventuellement la couleur des cheveux ou de la peau. Ils appliquent également une approche de soustraction de l'arrière-plan [Jojic 09] pour éviter que la répartition des couleurs ne soit contaminée par le décor et que le descripteur SCNCD n'en soit faussé. La figure 2.4 illustre les descripteurs obtenus sur un exemple avec et sans soustraction du décor.

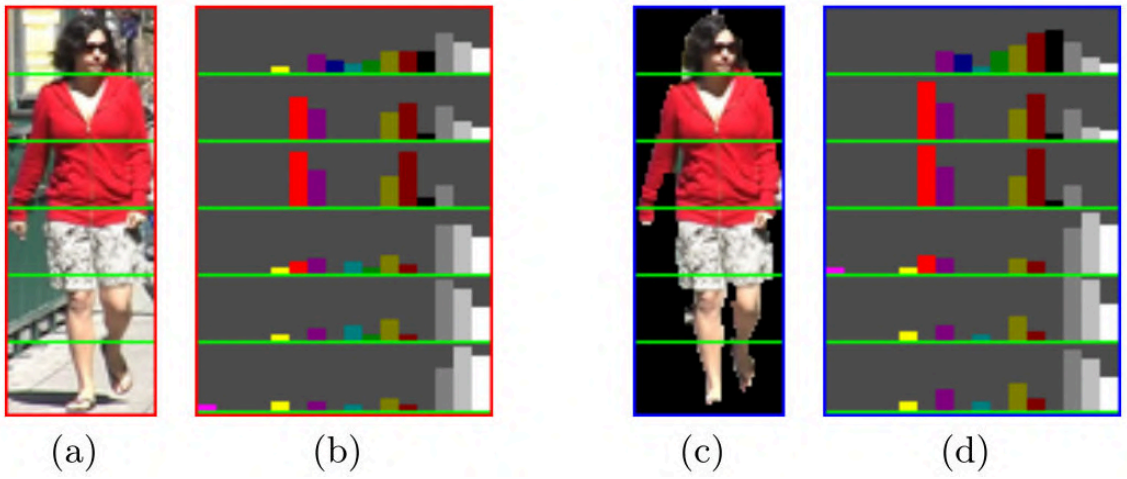


FIGURE 2.4 – Descripteur SCNCD [Yang 14] — (a) L'imagette d'une personne est découpée en 6 tranches horizontales de même taille. (b) Pour chaque tranche, on parcourt l'ensemble des pixels et on leur associe leur distribution des couleurs saillantes les plus proches pour obtenir la distribution de couleurs saillantes de la tranche. Le descripteur SCNCD est la concaténation des distributions des différentes tranches. (c) Une soustraction de l'arrière-plan est appliquée en plus du découpage en tranches. (d) Les représentations *Color Names* des tranches sans prise en compte de l'arrière-plan. Figure extraite de [Yang 14].

2.3.6 Descripteurs multipropriétés

Certains descripteurs combinent plusieurs caractéristiques à la fois. C'est le cas du descripteur LOMO (*LOcal Maximum Occurence*) [Liao 15] qui combine des aspects colorimétriques et de texture tout en étant multirésolution pour être invariant aux changements d'échelle. Les images sont également prétraitées par l'algorithme Retinex [Land 71] qui a tendance à rendre les couleurs plus vives, notamment sur les images sombres, rendant le descripteur robuste aux variations d'éclairage. Le descripteur se veut invariant aux différences de points de vue. Certains motifs de texture sont caractéristiques de certains types d'objets. Cependant, selon que l'objet est orienté vers la droite ou vers la gauche dans l'image, un tel motif peut apparaître en une position symétrique par rapport à l'axe vertical de l'objet. Pour rendre le descripteur robuste à ce type de difficultés, des histogrammes

de couleurs dans l'espace colorimétrique teinte-saturation-lumière ou *Hue Saturation Value* (HSV) sont calculés localement sur de patches extraits le long des tranches horizontales de l'image. Le descripteur LOMO intègre, pour chaque tranche horizontale, la valeur maximale de chaque intervalle des histogrammes de la tranche. La figure 2.5 illustre ce procédé.

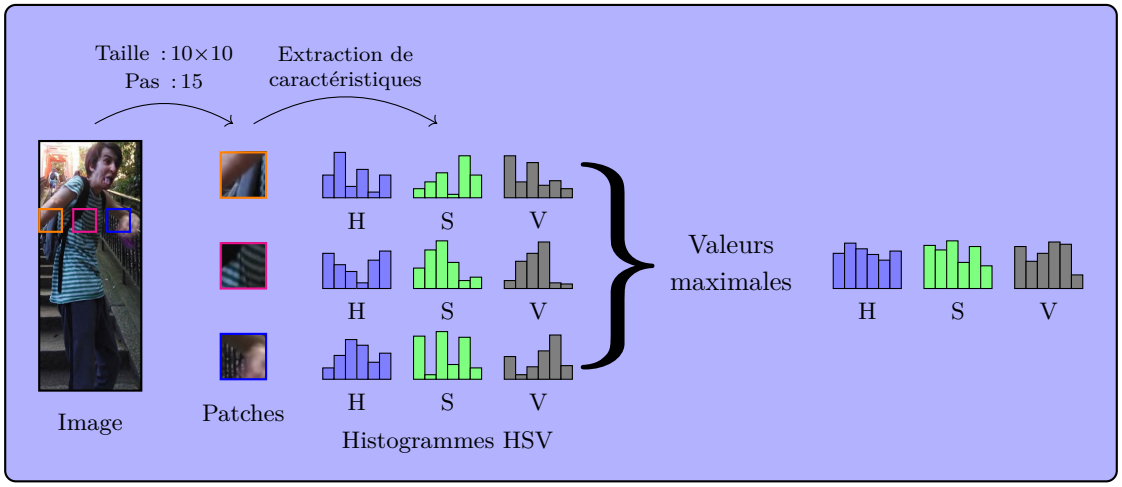


FIGURE 2.5 – Descripteur multipropriété LOMO [Liao 15] – Pour permettre une robustesse aux différences de points de vue, des histogrammes de couleurs HSV sont calculés localement le long des tranches horizontales de l'image et, pour chaque intervalle, seule la valeur maximale le long de la tranche est conservée dans le descripteur final.

2.3.7 Méthodes de détection par apprentissage profond

Dans cette partie, nous emploierons le vocabulaire spécifique à l'apprentissage profond. Les méthodes de détection seront présentées séparément des méthodes de réidentification. En effet, contrairement aux méthodes précédentes dans lesquelles nous avons pu isoler la partie description d'objet de la partie localisation d'objet, certaines approches utilisant l'apprentissage profond traitent les deux problèmes à la fois. D'autre part, la structure des réseaux diffère généralement entre les deux objectifs. De manière générale, les approches de vision reposant sur l'apprentissage profond utilisent des réseaux de neurones convolutifs (*Convolutional Neural Networks*, CNN). Deux types de méthodes se distinguent :

— Celles en deux étapes (*two shots*), dont les plus célèbres sont RCNN [Girshick 14], *Fast*-RCNN [Girshick 15] et *Faster*-RCNN [Ren 15], qui commencent par détecter des régions candidates de l'image, puis utilisent ces régions en entrée d'un CNN qui a été entraîné à reconnaître N catégories d'objets. La sortie du réseau est alors un vecteur de taille N indiquant, pour chaque catégorie d'objet, la probabilité que la région candidate contienne cette catégorie d'objet.

— Celles en une seule étape (*single shot*), dont les précurseurs sont YOLO (*You Only Look Once*) [Redmon 16] et SSD (*Single Shot Detector*) [Liu 16], qui utilisent direc-

tement l'image en entrée du CNN. Pour ce type de méthodes, le réseau fournit en sortie un ensemble de boîtes englobantes associées à des catégories d'objets.

Dans les approches en deux étapes, la première étape consiste à déterminer les régions de l'image ayant la plus grande probabilité de contenir un objet. Cette étape porte souvent le nom de recherche sélective. Nous pouvons citer [Felzenszwalb 04], dont s'inspire RCNN. L'algorithme commence par effectuer une segmentation initiale en groupant en régions les pixels voisins de niveaux de gris proches, puis regroupe itérativement les régions de niveaux de gris proches jusqu'à que tous les pixels soient groupés en une seule région. Des boîtes englobantes sont finalement obtenues à partir de toutes les régions des différents niveaux de segmentation, puis utilisées comme candidates pour la détection d'objets, comme présenté sur la figure 2.6. Cette approche multiéchelle permet de ne pas tester exhaustivement toutes les positions et tailles possibles de boîtes englobantes sur l'image, ce qui représente un important gain de temps.

FIGURE 2.6 – Exemple de recherche sélective [Felzenszwalb 04] – Première ligne : à partir d'une image d'entrée, une segmentation initiale est obtenue en regroupant les pixels aux niveaux de gris proches. Les régions obtenues sont itérativement regroupées. Deuxième ligne : à chaque niveau de segmentation, une boîte englobante candidate à la détection est formée pour chacune des régions. Les boîtes englobantes vertes correspondent aux détections conservées par l'algorithme et dans ce cas, ce sont celles attendues : celles qui contiennent une personne. Image issue de `https://www.geeksforgeeks.org/`.

L'approche RCNN [Girshick 14] utilise l'algorithme de recherche sélective décrit ci-dessus, en apportant une variante au niveau du regroupement des régions : à chaque itération, seules les deux régions voisines les plus similaires sont regroupées jusqu'à ce que l'image entière ne soit formée que d'une seule région. Le reste de l'approche est identique, à savoir l'initialisation et la génération des fenêtres candidates obtenues aux différents niveaux de segmentation. Chacune de ces fenêtres est ensuite redimensionnée à la même

taille, puis est utilisée en entrée du CNN décrit dans [Krizhevsky 12], entraîné pour reconnaître les 1000 classes du jeu de données ImageNet [Deng 09]. En sortie du réseau, chaque région candidate est alors décrite par un vecteur caractéristique issu de la dernière couche entièrement connectée du réseau (juste avant la couche de classification).

Bien qu'à sa parution, RCNN était l'approche qui donnait les meilleurs résultats parmi toutes les approches de détection existantes, le traitement se révèle relativement long, les auteurs rapportant des temps de calcul proches de la dizaine de secondes par image sur le jeu de données Pascal VOC 2010 [Everingham 10]. Plusieurs améliorations ont été proposées par la suite. La première, Fast-RCNN [Girshick 15], est plus rapide car, au lieu d'extraire les vecteurs caractéristiques du réseau indépendamment pour chaque région candidate, il extrait une matrice de caractéristiques sur l'ensemble de l'image en un seul passage dans le réseau, et réutilise cette matrice pour l'apprentissage d'un classifieur *Support Vector Machine* (SVM), dont nous rappellerons le fonctionnement dans le chapitre 4. Cela permet de grandement diminuer les temps de calcul de l'étape de prédiction. C'est désormais l'étape de recherche des régions d'intérêt, inchangée dans Fast-RCNN, qui s'avère être la plus longue. La deuxième amélioration, Faster-RCNN [Ren 15], s'attaque à ce problème en intégrant et en entraînant un réseau spécifique à la recherche des régions d'intérêt. Enfin, nous citons également Mask-RCNN [He 17], qui adapte Faster-RCNN pour obtenir une localisation des objets sous forme de régions, plus fine et détaillée qu'une simple boîte englobante.

La deuxième famille de méthodes de détection d'objets basées sur l'apprentissage profond est généralement qualifiée de *single shot* car elle utilise l'image entière en entrée d'un CNN sans recherche préalable de régions candidates, et renvoie un ensemble de boîtes englobantes correspondant aux objets détectés dans l'image. Utiliser un réseau de neurones directement sur une image pour y détecter les objets présents semble poser un problème : la structure en sortie d'un réseau a toujours la même dimension. Or, le nombre d'objets à détecter peut varier considérablement d'une image à une autre. La solution est alors de renvoyer un nombre fixe de boîtes englobantes, puis de les classer comme étant un objet ou non, et le cas échant, de prédire leur catégorie. C'est ce que fait notamment YOLO (*You Only Look Once*) [Redmon 16] qui prend en entrée une image qu'elle redimensionne puis découpe en cellules selon un quadrillage. Chaque cellule est « responsable » de la détection de B boîtes englobantes centrées en cette cellule, et chaque boîte englobante est représentée par un vecteur contenant les variables suivantes :

1. la probabilité p_c qu'une boîte englobante soit centrée sur la cellule courante c — dans le cas où cette probabilité vaut 0, la valeur des variables suivantes n'a aucune importance ;
2. les coordonnées du centre de la boîte englobante (b_x, b_y) et ses dimensions (b_w, b_h) ;
3. un ensemble de probabilités — autant que le nombre C de classes différentes que l'on veut pouvoir détecter — où chacune indique la probabilité que la boîte englobante

contienne un objet d'une classe donnée.

En annotant des images par des boîtes englobantes qui suivent cette structure, on peut alors entraîner le réseau à détecter différents types d'objets. YOLO a connu par la suite deux nouvelles versions, YOLOv2 [Redmon 17] et YOLOv3 [Redmon 18], apportant des modifications et des améliorations à la version originale, qui permettent notamment de détecter 9000 classes d'objets différentes et de prédire plus de boîtes englobantes par cellule.

Une autre approche de la famille des détecteurs *single shot* est le SSD (*Single Shot Detector*) [Liu 16]. Similaire à YOLO, cette approche se distingue par le fait qu'elle teste beaucoup plus de boîtes englobantes et à plusieurs résolutions. Le vecteur caractéristique est ainsi composé d'une agrégation des résultats de plusieurs filtres de convolution à différentes échelles pour chaque classe. De la même façon que pour YOLO, il est nécessaire d'annoter des images en suivant la structure des vecteurs caractéristiques pour ensuite entraîner le réseau.

2.3.8 Méthodes de réidentification par apprentissage profond

En réidentification, on suppose que l'on dispose déjà des imagettes de chaque objet. Contrairement à la détection, il n'y a donc pas besoin de localiser les objets. Plusieurs types de réseaux peuvent être utilisés. Comme pour les méthodes de détection par apprentissage profond en deux étapes, il est possible de donner chaque imagette en entrée d'un réseau entraîné sur un certain nombre de classes et d'en extraire sa **représentation latente** qui est le vecteur caractéristique issu de la dernière couche entièrement connectée du réseau (juste avant la couche de classification). On peut alors utiliser ce vecteur caractéristique comme un descripteur de l'imagette.

Il est possible d'utiliser des réseaux plus complexes tels que les réseaux siamois : il s'agit d'un réseau dédoublé prenant en entrée 2 imagettes, chacune passant dans un des réseaux et se rejoignant au niveau de la couche de classification en un seul neurone chargé d'apprendre à renvoyer 1 lorsque les imagettes représentent un même objet et -1 lorsque ce sont deux objets différents, comme illustré par la figure 2.7. L'entraînement du réseau fait appel à des triplets composés de deux images d'un même objet et d'une image d'un autre objet. Pour chaque image, le réseau apprend donc à partir d'un exemple positif et d'un exemple négatif. Pour les réseaux siamois, deux grandes familles d'approches se distinguent :

1. les approches fondées sur l'**apprentissage de la représentation**, utilisées notamment dans [Sunderrajan 15], qui, pour une mesure donnée, cherchent à **apprendre le descripteur** qui minimise la distance entre paires d'imagettes d'un même objet et maximise la distance entre paires d'imagettes d'objets différents ;

2. les approches fondées sur l'**apprentissage de la mesure** qui, pour un descripteur donné, cherchent à **apprendre la mesure** qui minimise la distance entre paires d'imagettes d'un même objet et maximise la distance entre paires d'imagettes d'objets différents, comme proposé dans [Varior 16].

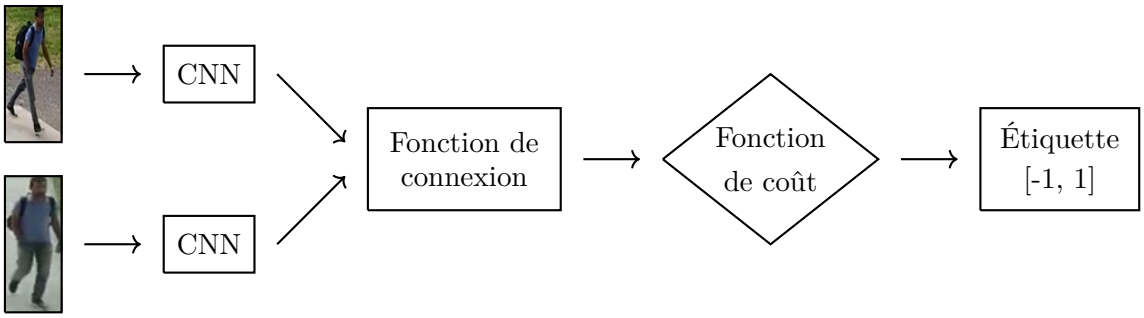


FIGURE 2.7 – Chaîne de traitement de la réidentification basée sur des réseaux siamois — Une paire d'imagettes est donnée en entrée de deux réseaux identiques. Des caractéristiques sont extraites de ces réseaux puis combinées et soumises à un fonction de coût assignant une valeur censée être proche de 1 pour les paires d'imagettes d'un même objet et proche de -1 pour des paires d'imagettes d'objets différents.

Lorsque l'on dispose de plusieurs imagettes d'un objet, comme par exemple lors du suivi d'objets dans une vidéo, cela peut aider la tâche de réidentification. Différentes stratégies peuvent être envisagées pour exploiter les multiples imagettes : on peut par exemple calculer la plus petite distance parmi toutes les paires d'imagettes de deux objets, ou encore la distance moyenne. C'est notamment ces stratégies que nous avons adoptées dans la publication [Roman-Jimenez 20], menée en collaboration avec d'autres équipes impliquées dans le projet VICTORIA, et portant sur la réidentification de véhicules. Nous y caractérisons chaque imagette par sa représentation latente, comme illustré par la figure 2.8, puis nous comparons les représentations latentes des imagettes multiples de différents objets, voir figure 2.9. L'avantage d'utiliser des représentations latentes comme descripteurs est qu'elles sont faciles à extraire et à comparer. Le réseau peut également être modifié sans que cela n'impacte la structure de la représentation latente ou la mesure à utiliser. Dans cet article, nous testons plusieurs architectures de réseau avec 3 mesures différentes. Nous comparons les performances de trois types de scénarios selon que l'on dispose d'une seule ou de plusieurs images du véhicule requête et d'une ou de plusieurs images des véhicules de la base d'objets : les scénarios *image-to-image*, *image-to-track* et *track-to-track*. Nous montrons que la distance cosinus avec des comparaisons *track-to-track* en choisissant la distance minimale parmi toutes les paires d'images donne les meilleures performances. Nous avons choisi de ne pas détailler davantage ce travail dans le corps du mémoire car il s'agit d'un travail collaboratif mené par tous les membres de l'IRIT impliqués dans le projet VICTORIA et il dépasse le simple cadre de cette thèse. Toutefois, au chapitre 6, lors du bilan des contributions, nous apporterons plus de détails.

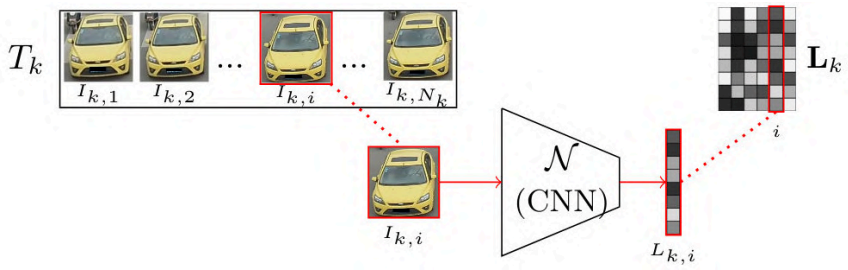


FIGURE 2.8 – Extraction des caractéristiques latentes d'imagettes multiples — Chaque imagette est donnée en entrée d'un CNN et le vecteur caractéristique issu de l'avant-dernière couche du réseau est extraite pour servir de descripteur, appelé descripteur latent. Ces descripteurs latents sont ensuite rassemblés dans une matrice. Figure extraite de [Roman-Jimenez 20].

2.3.9 Conclusion sur les différents descripteurs

Nous avons présenté les différents types de descripteurs les plus répandus de la bibliographie :

- ceux s'appuyant sur une propriété particulière telle que la couleur [Yang 14] ou les contours [Dalal 05], simples mais peu robustes aux cas difficiles ;
- ceux qui combinent plusieurs propriétés [Liao 15] et permettent de caractériser l'objet de façon plus riche mais qui demandent plus de calculs ;
- ceux basés sur des masques [Viola 01] qui nécessitent de faire glisser des masques à différentes échelles en de nombreuses positions de l'image ;
- ceux rigides par parties [Felzenszwalb 05] qui permettent de tenir compte de l'aspect déformable de certains types d'objets en décrivant différentes parties de l'objet.

Nous avons également présenté les méthodes reposant sur l'apprentissage profond. Pour la détection, nous distinguons celles en une étape [Redmon 16, Liu 16] de celles en deux étapes [Girshick 14, Girshick 15, Ren 15]. Pour la réidentification, nous distinguons celles basées sur des réseaux siamois que ce soit par apprentissage de la mesure [Varior 16] ou par apprentissage de la représentation [Sunderrajan 15], et celles qui extraient des représentations latentes et s'en servent comme descripteurs [Roman-Jimenez 20]. Notre choix s'est porté sur ces approches en particulier de par leur diversité et leur simplicité d'utilisation. Dans nos contributions, nous emploierons ces différentes méthodes et évaluerons les résultats qu'elles permettent d'obtenir.

2.4 Traitement de l'arrière-plan

L'arrière-plan, ou le décor, fait partie intégrante des vidéos. Pour une caméra fixe, il se caractérise généralement par le fait qu'il varie peu au cours du temps. D'autre part,

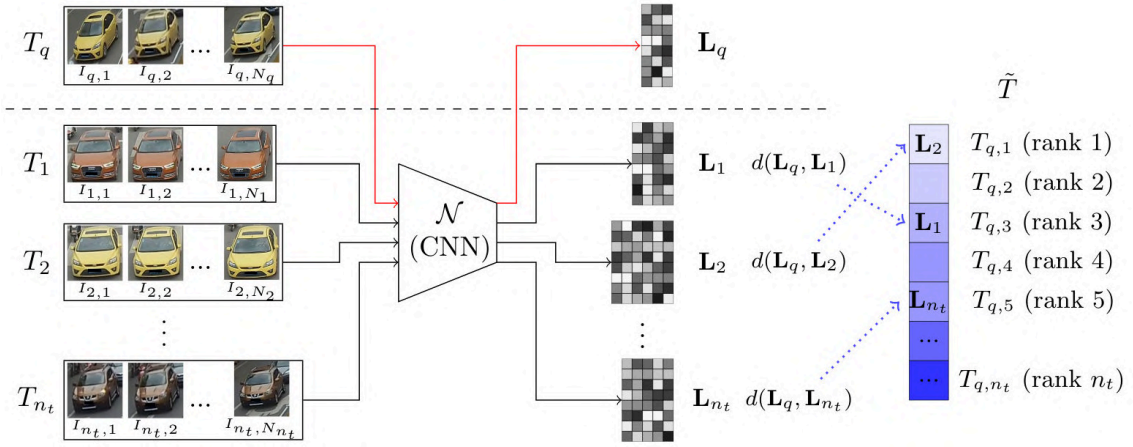


FIGURE 2.9 – Réidentification par imagettes multiples — T_q représente des imagettes multiples d'un même objet requête et $T_1, T_2, \ldots, T_{n_t}$ sont des imagettes multiples correspondant à différents objets. Les matrices de descripteurs latents $L_1, L_2, \ldots, L_{n_t}$ sont extraites et comparées à la matrice de descripteurs latents L_q pour fournir un classement des imagettes multiples par ordre de similarité avec la requête. Figure extraite de [Roman-Jimenez 20].

des entités statiques peuvent être considérées comme faisant partie du décor. Ainsi, on considèrera qu'une personne immobile ou un véhicule garé font partie du décor. Lorsque l'on cherche un lien entre deux vidéos, les éléments de décor sont prédominants à l'écran. Ainsi, dans certains cas, on peut être en mesure de déterminer qu'il y a ou non un lien simplement en étudiant le décor.

La représentation du décor est un sujet d'étude à part entière. Comme pour la détection et la réidentification, cette thèse s'appuie sur des approches de modélisation de l'arrière-plan sans apporter de contribution au domaine. Nous nous contenterons d'expliciter ses principes généraux en présentant succinctement quelques approches. Nous distinguons deux familles de méthodes :

— les **méthodes de modélisation du décor par pixel** qui attribue à chaque pixel un modèle de décor ;

— les **méthodes de modélisation globale du décor** qui fournissent une segmentation sémantique de l'image.

Dans la catégorie des méthodes de modélisation du décor par pixel, chaque pixel dispose de son propre modèle de décor. Généralement, chaque pixel est caractérisé par une ou plusieurs couleurs qu'il prend lorsqu'il n'est pas occupé par un objet mobile. Par exemple, on peut considérer un modèle de 20 couleurs par pixel [Russell 06]. Par la suite, on compare les pixels de chaque nouvelle image aux couleurs de son modèle de décor pour déterminer s'il fait partie ou non du décor. Si la couleur du pixel est proche d'au moins 2 des 20 couleurs de son modèle, le pixel est considéré comme faisant partie du décor. L'initialisation se fait

sur quelques images, soit prises en début de vidéo, soit prises aléatoirement dans la vidéo. Cette étape est délicate, car il est possible que des objets soient présents sur les images choisies. Dans le cas d'un objet immobile pendant toute l'initialisation et qui disparaît après, les pixels situés au niveau de la zone où il se trouvait donnent des fausses alarmes. Pour éviter ce phénomène, le modèle de décor des pixels est régulièrement mis à jour en modifiant aléatoirement l'une des 20 couleurs du modèle de certains pixels. Un exemple de modèle d'arrière-plan obtenu par l'approche [Russell 06] est illustré par la figure 2.10.

FIGURE 2.10 – Obtention d'un modèle d'arrière-plan par l'approche décrite dans [Russell 06] — À partir de 20 images initiales (4 sont présentées à gauche), un modèle d'arrière-plan est appris pour chaque pixel. Figure extraite de [Russell 06].

Nous pouvons également citer [Guyon 12] qui présente différentes approches de soustraction de décor qui construisent un modèle de décor dans un espace de faible dimension en utilisant une analyse en composantes principales.

Dans les approches de modélisation globale du décor par segmentation, le but est d'obtenir une segmentation de l'image où chaque pixel correspond à une classe parmi plusieurs types de décors (par exemple herbe, feuillages ou bâtiment). C'est en cela que la segmentation est sémantique : chaque région correspond à un type de décor, chaque région a une classe sémantique. Ces approches font généralement appel à de l'apprentissage profond, par exemple [Zhu 19, Reda 18]. Le réseau est entraîné sur des images segmentées manuellement pour apprendre à fournir une segmentation sur d'autres images. Un exemple de résultat obtenu par la méthode de segmentation sémantique proposée dans [Zhu 19] est présenté sur la figure 2.11. Nous avons utilisé le code mis à disposition en ligne [1] par les auteurs.

Maintenant que nous avons présenté les outils essentiels pour décrire et analyser une vidéo, en distinguant les objets d'intérêt du décor de la scène, nous pouvons aborder les travaux existants dans le domaine plus précis dans lequel se situe cette thèse : l'estimation des liens entre vidéos.

1. https://github.com/NVIDIA/semantic-segmentation

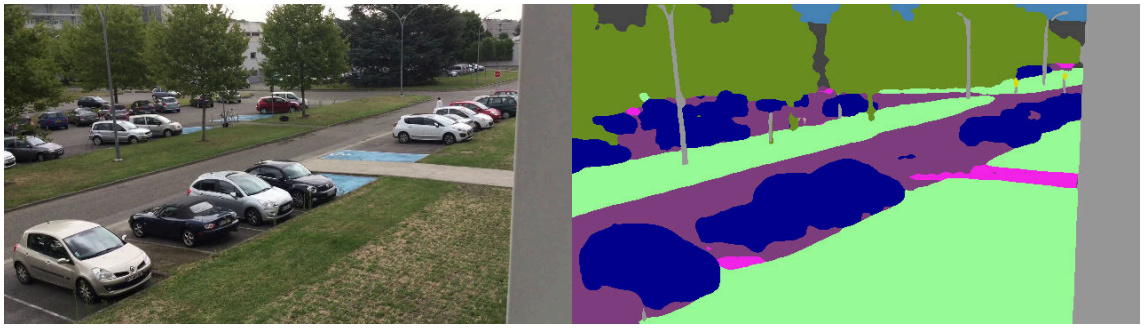


FIGURE 2.11 – Segmentation sémantique d'une image par l'algorithme de [Zhu 19] — À gauche, une image issue du jeu de données que nous avons proposé et que nous présentons au chapitre 3. À droite, la segmentation sémantique de cette image, chaque couleur correspondant à un type de décor : la route apparaît en mauve, les zones piétons en rose, l'herbe en vert pâle, la végétation en vert foncé, les bâtiments en gris et les véhicules en bleu.

2.5 Estimation des liens entre vidéos

2.5.1 Liens entre vidéos sans recouvrement

Dans le cas sans recouvrement, l'objectif est de construire un graphe de liens entre caméras qui permette de suivre le déplacement d'un objet, généralement un piéton ou un véhicule, entre les caméras du réseau. Dans ce graphe, chaque caméra est représentée par un nœud et les liens entre caméras sont représentés par des arc ou des arêtes (orientés ou pas) qui peuvent être pondérées par la probabilité qu'une entité passe de l'une à l'autre caméra. Un lien entre deux caméras C_1 et C_2 s'interprète alors comme suit : « un objet sortant du champ de vue de C_1 (respectivement C_2) est susceptible d'apparaître dans C_2 (respectivement C_1) après un certain temps T ». Ce graphe apporte ainsi des informations sur la topologie du réseau de caméras.

Afin d'obtenir ce graphe, de nombreuses approches déterminent, pour chaque vidéo, les zones de transit ou zones d'entrée-sortie : il s'agit de régions, souvent situées au niveau des bords de l'image, dans lesquelles apparaissent ou disparaissent les objets. Pour ce faire, la chaîne de traitement, présentée sur la figure 2.12, comporte les étapes suivantes :

— déterminer les trajectoires des objets apparaissant dans chaque vidéo, généralement par des méthodes de suivi d'objets ;

— extraire l'ensemble des points de début et de fin de toutes les trajectoires et appliquer un algorithme de regroupement tel que k-moyennes [MacQueen 67] ou *mean-shift* [Comaniciu 02] ;

— pour chaque groupe de points, déterminer la distribution des points dans la région, par exemple un modèle gaussien.

Après avoir appliqué ces étapes à chaque vidéo individuellement, on dispose de l'ensemble des zones d'entrée-sortie. Il reste alors à déterminer quelles paires de régions (R_1, R_2) d'entrée-sortie issues de vidéos différentes sont liées. Si deux zones sont liées, on peut s'attendre à ce que des objets disparaissant de l'une apparaissent dans l'autre après un certain délai. Comme les champs de vues ne se recouvrent pas, il existe généralement des zones dans la scène où l'on ne peut pas savoir ce qui se passe : l'objet peut s'y arrêter, changer d'allure ou même faire demi-tour. Pour traiter cette difficulté, les auteurs de [Makris 04] dressent la distribution des délais entre les temps de chaque point de fin de trajectoire situé dans R_1 (respectivement R_2) et de chaque point de début de trajectoire situé dans R_2 (respectivement R_1). L'idée est alors qu'en moyenne, si les zones sont liées, les objets vont avoir tendance à passer d'une zone à l'autre avec des délais proches, ce qui se traduit par l'existence d'un mode dans la distribution des délais. À l'inverse, lorsqu'aucun délai ne se distingue des autres, les zones sont considérées comme non liées. Pour éviter de prendre en compte tous les délais entre zones d'entrée-sortie de toutes les paires d'objets différents, les auteurs de [Gilbert 08] ne tiennent compte que des délais entre paires d'objets respectant un critère de similarité. Celui-ci s'appuie sur la couleur, la taille relative et le mouvement.

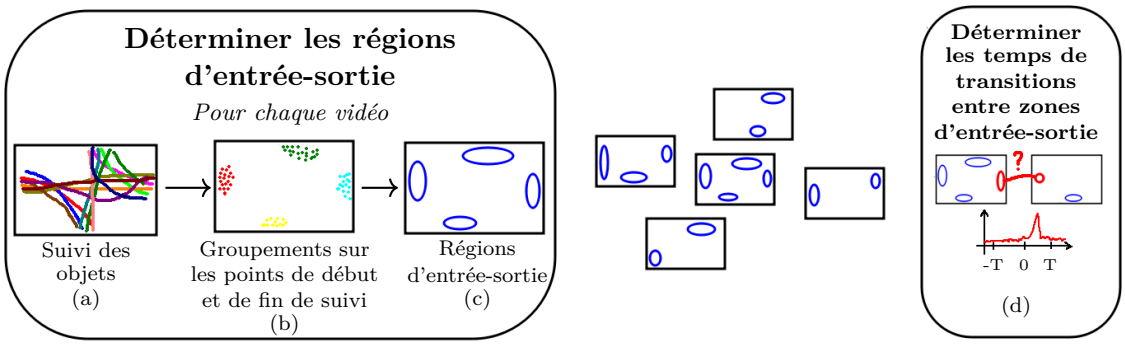


FIGURE 2.12 – Chaîne de traitement pour déterminer les liens d'un réseau de caméras sans recouvrement — Dans chaque vidéo, après avoir déterminé les trajectoires de chaque objet (a), un algorithme de regroupement est appliqué sur les points de début et de fin des trajectoires (b) et chaque groupe de points définit une région d'entrée-sortie (c). Les distributions de délais entre apparitions et disparitions d'objets au sein de zones d'entrée-sortie issues de vidéos différentes sont calculées pour déterminer les éventuels liens entre ces zones (d).

Dans [Chen 11b], les auteurs remarquent que certains liens, bien que valides, ont peu d'intérêt et peuvent même prêter à confusion pour des opérateurs humains. Pour comprendre ce que cela signifie, nous reprenons l'exemple qu'ils donnent dans leur article et qui s'appuie sur un réseau de trois caméras C_1, C_2 et C_3 placées dans cet ordre le long d'un couloir, avec des champs de vue sans recouvrement orientés perpendiculairement au sens de la marche, de la même façon que les caméras C_3, C_4 et C_5 de la figure 2.1. Les

méthodes décrites jusqu'ici devraient trouver le lien entre C_1 et C_2 avec un délai moyen τ_{12}, le lien entre C_2 et C_3 avec un délai moyen τ_{23}, mais également un troisième lien entre C_1 et C_3 avec un délai τ_{13}. On peut noter que $\tau_{13} > \tau_{12} + \tau_{23}$ car pour passer de C_1 à C_3, il faut passer d'une zone de sortie de C_1 à une zone d'entrée de C_2 (τ_{12}), puis traverser C_2, puis passer d'une zone de sortie de C_2 à une zone d'entrée de C_3. Ce lien supplémentaire est bien conforme à l'interprétation formulée plus haut : un objet sortant du champ de vue de C_1 est susceptible d'apparaître plus tard dans C_3. Pour autant, les auteurs qualifient ce lien de « faible » et cherchent à le supprimer pour plusieurs raisons :

— il augmente la complexité du réseau et donc les temps de traitement ;

— il augmente le taux d'erreurs de suivi entre caméras : les objets aperçus dans C_1 qui n'apparaissent pas ensuite dans C_2 n'apparaîtront pas non plus dans C_3 ;

— dans un cadre d'utilisation par des opérateurs humains qui souhaiteraient suivre une personne aperçue dans C_1, ce type de lien augmenterait le nombre de candidats proposés en indiquant C_2 et C_3.

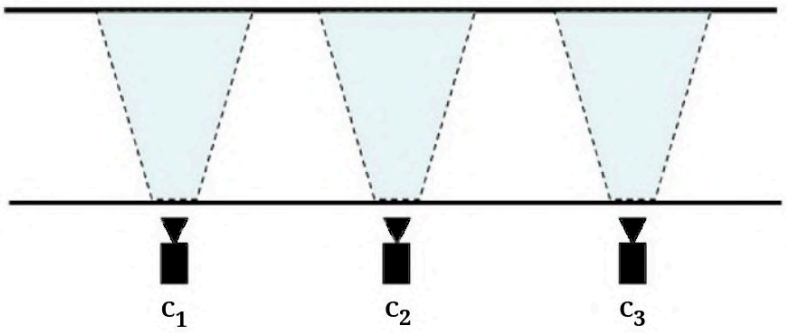


FIGURE 2.13 – Illustration d'un lien faible — Chaque personne traversant le couloir passe devant les trois caméras successivement. Les approches décrites précédemment permettent alors de trouver des liens entre chaque paire de caméras. Cependant, le lien entre C_1 et C_3 est un lien qualifié de « faible » car il omet le passage par C_2.

Pour supprimer les liens faibles, les auteurs s'appuient sur deux propriétés en présence d'un lien faible : l'existence d'un cycle dans le graphe de liens et le fait, évoqué plus haut, que le délai d'un lien faible entre deux caméras C_1 et C_N est supérieur à la somme des délais d'une succession de liens menant de C_1 à C_N. La figure 2.14 illustre les résultats obtenus sans suppression des liens faibles par l'approche [Makris 04], et avec suppression des liens faibles par l'approche [Chen 11b]. Le graphe obtenu après cette suppression rend mieux compte de la topologie du réseau de caméras.

Pour traiter les cas où les résultats de suivi des objets ne sont pas fiables, notamment lorsque la cadence [2] de la caméra est faible, les auteurs de [Loy 09] proposent une approche basée sur la corrélation des profils d'activité au sein de différentes régions des vidéos. Ils

2. Il s'agit du nombre d'images acquises par seconde.

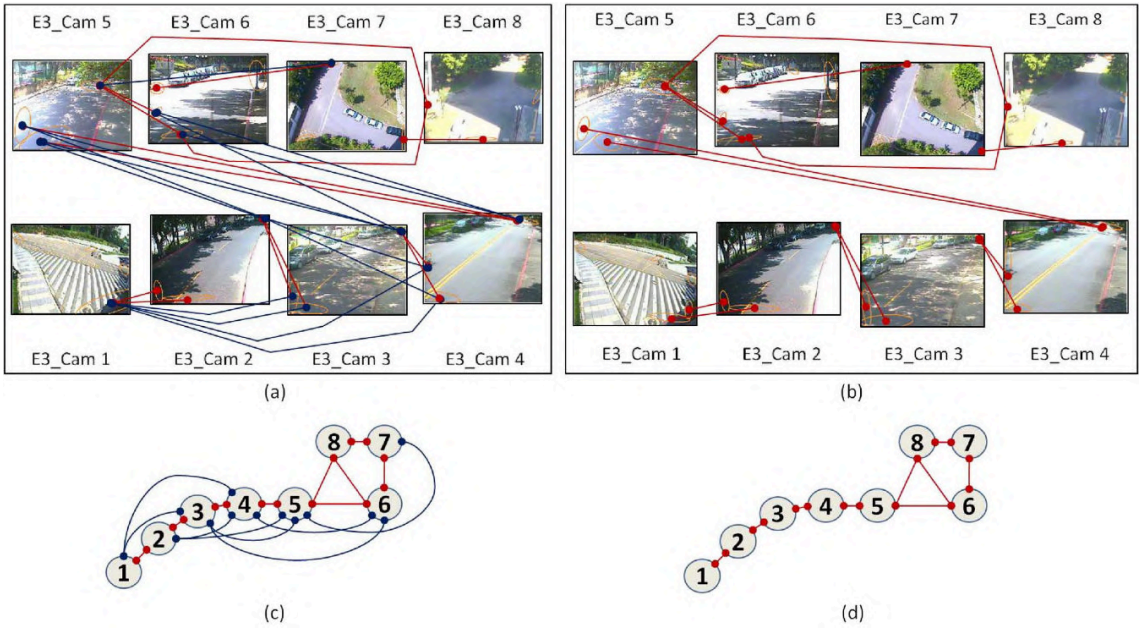


FIGURE 2.14 – Exemple de graphes de liens avec et sans suppression des liens faibles — Liens entre régions d'entrée-sortie sur un ensemble de vidéos par l'approche [Makris 04] (a) et par l'approche [Chen 11b] (b). Les graphes de liens respectifs sont représentés en dessous (c) et (d). Les liens faibles sont représentés en bleu et les liens valides en rouge. Figure extraite de [Chen 11b].

commencent par définir un modèle d'arrière-plan en s'inspirant de [Russell 06], pour être en mesure de déterminer, pour chaque pixel, s'il fait partie de l'arrière-plan, donc du décor, ou du premier plan, donc d'un objet, comme décrit dans la section 2.4. Une fois le modèle de décor initialisé, les auteurs découpent la fenêtre en blocs et caractérisent chaque bloc par son taux d'occupation par des éléments au premier plan en mouvement et par des éléments au premier plan immobiles au cours du temps. Après avoir déterminé si un pixel fait ou non partie du premier plan en comparant son code RGB à ceux de son modèle de décor, la distinction entre élément en mouvement et élément immobile se fait par une différence entre images successives. Les blocs spatialement proches présentant des profils d'occupation similaires sont regroupés, formant un découpage du cadre en régions d'activité. Ces régions sont ensuite utilisées de la même façon que les zones d'entrée-sortie en cherchant l'existence éventuelle d'un délai entre deux régions d'activité issues de deux caméras différentes. La chaîne de traitements de l'approche est illustrée par la figure 2.15.

Enfin, les auteurs de [Cho 19] proposent de conjointement réidentifier des personnes tout en déterminant les liens entre caméras. En partant des réidentifications les plus fiables, ils déterminent des premières zones d'entrée-sortie entre caméras. Une fois ces zones initialisées, ils renforcent ensuite itérativement la réidentification en s'appuyant sur la topologie et inversement. À chaque itération, de nouvelles réidentifications sont ainsi trouvées ainsi que de nouvelles zones d'entrée-sortie.

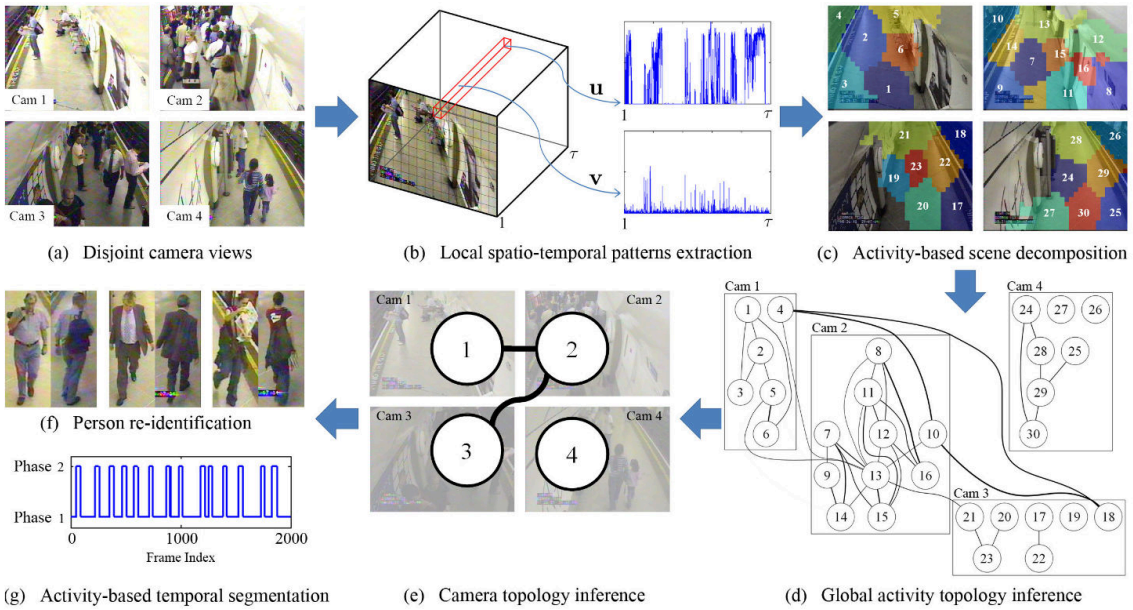


FIGURE 2.15 – Chaîne de traitements de l'approche décrite dans [Loy 09] — Des vidéos sans recouvrement de champs de vue (a) sont découpées en cellules et chaque cellule est caractérisée par le profil d'activité d'éléments au premier plan (b). Les cellules proches aux profils d'activité similaires sont regroupés en régions (c) puis les liens entre régions sont estimés (d). La topologie du réseau de caméras est alors déduite (e) et aide à la tâche de réidentification (f et g). Figure extraite de [Loy 09].

2.5.2 Liens entre vidéos avec recouvrement

Dans le cas avec recouvrement des champs de vue, le graphe de liens a un sens différent. Les nœuds représentent toujours les caméras, mais cette fois, deux caméras sont liées si leurs champs de vue se croisent, c'est-à-dire si elles observent, au moins partiellement, un même endroit. Dans certaines approches, on cherche également à expliciter les correspondances spatiales entre caméras.

Dans le cas d'une scène plane, les auteurs de [Stauffer 03] proposent de déterminer une transformation homographique permettant de passer d'un point d'une caméra à son correspondant dans une autre caméra. Plus de détails sur les homographies sont donnés en annexe B. Ils disposent d'un ensemble de trajectoires d'objets dans différentes vues, mais ne connaissent pas les correspondances entre trajectoires de vues différentes. Pour chaque paire de vidéos (V_1, V_2), ils estiment la transformation homographique qui fait se correspondre le plus de trajectoires de V_1 avec une trajectoire de V_2. Chaque trajectoire n'étant pas visible dans son intégralité dans chaque vue, ils restreignent chaque paire de trajectoires à l'intervalle de temps où elles sont visibles à la fois dans V_1 et dans V_2.

Dans un cas plus général, à partir d'un ensemble de vidéos issu d'un réseau de caméras, les auteurs de [Van Den Hengel 06] découpent chaque vidéo en cellules selon une grille régulière et cherchent à construire un graphe de liens qui présente les correspondances

entre cellules. Ils partent pour cela d'une situation où toutes les paires de cellules issues de vidéos différentes sont liées puis cherchent à supprimer les liens erronés. Pour cette tâche, ils définissent le principe d'exclusion qui se base sur l'observation suivante : « Si le champ de vue d'une caméra est occupé et que le champ de vue d'une autre caméra est simultanément non occupé, alors les deux caméras ne peuvent pas observer le même endroit ». Ils choisissent d'appliquer ce principe à l'échelle des cellules. Appliquer rigoureusement le principe d'exclusion signifierait rompre le lien entre deux cellules à la moindre occurrence d'une exclusion, c'est-à-dire la présence d'un objet dans l'une des cellules tandis qu'aucun objet n'est présent dans l'autre cellule. En pratique, de nombreuses imprécisions risquent de compromettre l'approche que ce soit au niveau de la détection d'objets et de sa position, ainsi que temporellement : les repères temporels des vidéos peuvent être légèrement décalés. Les auteurs apportent alors plusieurs modifications pour prendre en compte ces problèmes et rendre leur approche plus robuste. Ainsi, pour une paire de cellules (c_i, c_j) une exclusion est avérée si un objet est détecté dans c_i au temps t, alors qu'aucun objet n'est détecté dans c_j ou dans son voisinage spatial — les 8 cellules autour de c_j — que ce soit au temps t ou proche du temps t. Enfin, la proportion d'exclusions avérées doit être supérieure à un certain seuil pour que le lien entre deux cellules soit effectivement supprimé. Par la suite, les auteurs de [Detmold 07] ont rapporté des résultats portant sur un réseau de 83 caméras réparties sur un campus universitaire sur une période de deux heures. Les liens établis par leur approche entre certaines vues sont présentés sur la figure 2.16.

Les méthodes présentées dans cette partie donnent un aperçu de ce qu'il est possible de faire lorsque l'on cherche des liens entre vidéos sans aucune connaissance a priori. L'idée proposée dans [Loy 09] qui consiste à caractériser les régions d'une vidéo par un profil d'activité et qui a été utilisée pour traiter des réseaux de caméras sans recouvrement pourrait être exploitée dans le cas avec recouvrement. Au lieu de chercher des régions proches aux profils d'activité similaires au sein d'une même vidéo, des régions aux profils d'activité similaires dans des vidéos différentes sont un indice intéressant de correspondance entre les régions. Nous pouvons également employer le principe d'exclusion décrit dans [Van Den Hengel 06]. Notre première contribution exploite ces deux approches.

2.6 Description des vidéos par leurs histoires

Dans cette partie, nous présentons notre première contribution qui porte sur la recherche de liens dans une collection de vidéos. Lorsque deux vidéos aux champs de vue en recouvrement sont liées, il existe des régions issues de ces deux vidéos qui observent la même chose et il est probable qu'on puisse y observer simultanément les mêmes objets. Nous exploitons cette idée et proposons un descripteur spatio-temporel multirésolution résumant le contenu d'une vidéo. Chaque vidéo est d'abord traitée individuellement : à intervalles

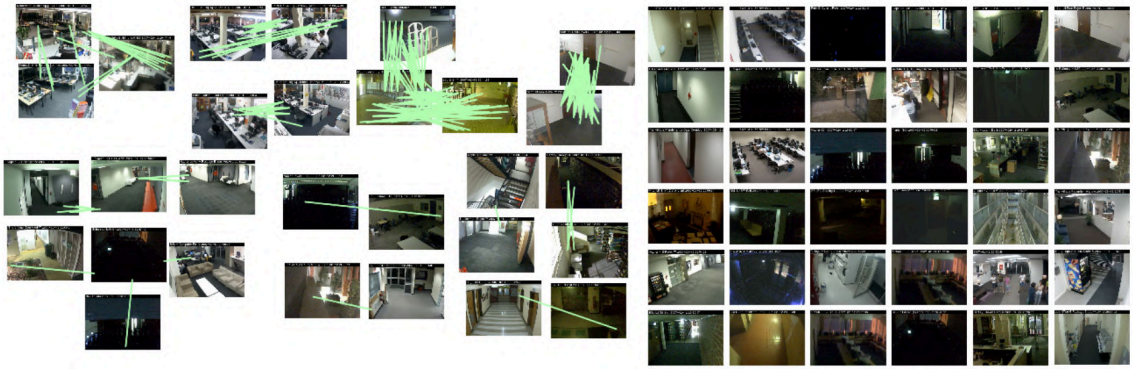


FIGURE 2.16 – Exemple de graphe de liens avec recouvrement — À partir d'un ensemble de caméras, des liens spatiaux sont établis entre les vues et représentés par des segments verts. À droite, un ensemble de vues où aucun lien n'a été trouvé. Les vues ont été disposées manuellement pour regrouper celles où du recouvrement a été trouvé. Résultats obtenus et présentés dans [Detmold 07].

réguliers, les objets présents sont détectés sous forme d'une boîte englobante et d'une catégorie, puis nous les caractérisons avec des descripteurs d'apparence parmi ceux présentés dans ce chapitre. La vidéo est ensuite découpée en cellules à différentes résolutions et nous assignons à chaque cellule ce que nous appelons son histoire : il s'agit de la liste des objets qui y ont été détectés au cours du temps, chaque objet étant représenté par une catégorie et un descripteur d'apparence. Cette notion est un concept clé dans notre approche de construction du graphe de liens entre vidéos. Les liens entre cellules de différentes vidéos sont alors déterminés en comparant leurs histoires : deux cellules se correspondent si, au cours du temps, elles présentent des détections simultanées d'objets de la même catégorie avec des apparences similaires. Nous définissons pour cela une mesure de dissemblance entre histoires, d'autant plus faible que des objets de même catégorie et d'apparences proches y sont simultanément détectés au cours du temps. Les paires de vidéos avec suffisamment de cellules en correspondance sont alors considérées comme ayant des champs de vue en recouvrement et sont donc liées. L'utilisation de la multirésolution avec différentes échelles spatiales et temporelles permet d'éviter de comparer de nombreuses paires de régions de résolution fine lorsque la concomitance des détections a déjà été établie à une résolution plus grossière. À l'inverse, elle permet d'éviter d'effectuer des recherches à des résolutions plus fines dès que la dissemblance ne fait plus de doute. La figure 2.17 illustre notre approche. Les expérimentations sont présentées dans le chapitre 3.

2.6.1 Terminologie employée et définition

Lorsque l'on raconte une histoire, on décrit généralement les personnages, leurs apparences, les lieux qu'ils traversent et leurs actions, avec plus ou moins de détails, et dans l'ordre chronologique. Pour ces raisons, nous avons choisi le terme générique « histoire »

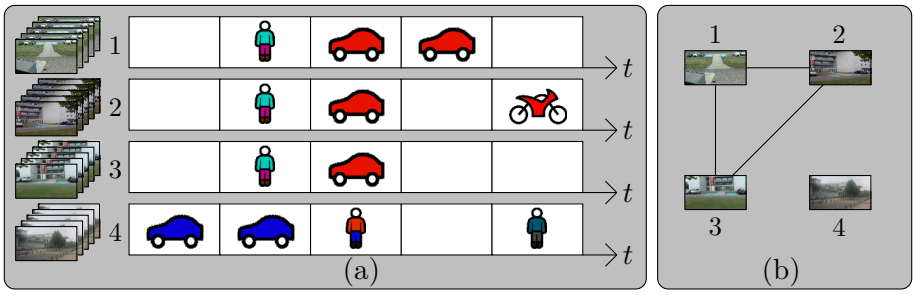


FIGURE 2.17 – Approche proposée pour la construction d'un graphe de liens – Pour chaque vidéo d'un ensemble de vidéos provenant de caméras statiques, les objets présents sont détectés pour formuler les *histoires* des vidéos (a), puis des liens sont formés entre les vidéos aux histoires similaires pour construire un graphe de liens (b) où chaque arête indique une paire de vidéos avec des champs de vue qui se recouvrent.

pour regrouper des propriétés variées telles que la catégorie, la position et l'apparence d'objets détectés dans une vidéo au fil du temps.

Considérons que chaque vidéo est découpée en régions, par exemple en cellules selon un quadrillage régulier. Pour calculer l'histoire d'une région, des objets y sont régulièrement détectés aux instants $\tau_i = \tau \times i$ par des algorithmes de détection de l'état de l'art parmi ceux cités dans ce chapitre. Nous n'utilisons donc pas de méthode de suivi des objets, seulement des boîtes englobantes détectées à intervalles réguliers. Ainsi, pour déterminer qu'un objet est statique, nous regarderons le recouvrement entre les boîtes englobantes détectées au temps τ_i et celles détectées au temps τ_{i-1}. L'histoire de la région consiste alors en une liste temporelle qui, à chaque instant τ_i, contient l'ensemble des objets détectés dans cette région, comme illustré par la figure 2.18. Nous parlons de l'histoire de la vidéo lorsque la région considérée occupe tout le cadre.

Nous assignons chaque détection à la cellule qui contient son **point de contact au sol**, c'est-à-dire le pixel situé au milieu de l'arête horizontale inférieure de sa boîte englobante, comme dans [Khan 03] (voir figure 2.19). Ce choix est motivé par le fait que cela ancre le pixel dans le plan du sol, ce qui permet d'éviter des ambiguïtés dues à la profondeur : de manière générale, un même pixel peut être occupé par des objets situés à des distances différentes de la caméra, mais, dans ce cas, leurs pixels au niveau du sol sont différents.

L'histoire d'une région R au cours de l'intervalle de temps $\tau_0, ..., \tau_i$ est notée $S_R^{0 \to i}$. L'histoire en un instant τ_j est notée S_R^j et l'histoire sur toute la durée de la vidéo est notée S_R. Nous notons V la région correspondant au cadre complet et T le nombre total d'instants τ_i. Cette définition permet de constituer des histoires à différentes résolutions spatiales et temporelles en ne considérant que les objets détectés dans un certain intervalle de temps et dans une certaine région du cadre (voir figure 2.20). L'utilisation d'une analyse multirésolution permet d'estimer les liens entre régions de façon robuste et de réduire les temps de calculs, comme détaillé dans la section 2.6.3. La prochaine étape que nous

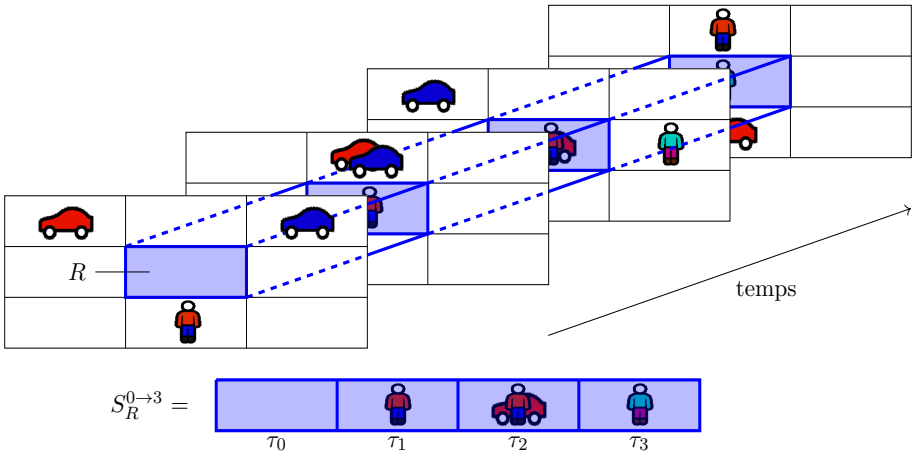


FIGURE 2.18 – Histoire d'une région – À des pas de temps réguliers, les objets sont détectés dans chaque région de la vidéo et se font assigner une catégorie et un descripteur d'apparence. L'histoire d'une région donnée R apparaît en bleu. Elle raconte qu'au temps τ_0, il n'y a aucun objet dans R, puis qu'à τ_1, il y a une personne, et ainsi de suite.

décrivons consiste à définir une distance entre histoires.

2.6.2 Distance entre histoires

Afin de définir la distance entre deux histoires de même longueur S_R et $S_{R'}$, nous introduisons d'abord la notion d'objet commun aux deux histoires. Un objet O est commun aux deux histoires s'il apparaît dans la région R de la première vidéo au temps τ_i et si un objet O' de même catégorie que O et d'apparence similaire apparaît dans la région R' ou dans son voisinage spatial (par exemple, dans son 8-voisinage), au temps τ_i ou dans son voisinage temporel $\{\tau_{i-k},\cdots,\tau_i,\cdots,\tau_{i+k}\}$, comme illustré par la figure 2.21. La similarité d'apparence est établie si la distance entre les descripteurs d'apparence est inférieure au seuil d'apparence σ_{app}. On dit alors que O a trouvé un correspondant dans R'.

Lors du processus d'analyse des vidéos, de nombreuses difficultés peuvent intervenir :

— certaines détections d'objets peuvent être manquantes ou, à l'inverse, erronées ;

— certains objets peuvent être occultés ;

— des correspondances peuvent être trouvées entre objets différents.

Pour ces raisons, il est logique de ne pas chercher à respecter la contrainte d'unicité, c'est-à-dire à faire correspondre à chaque objet détecté dans une vidéo un et un seul objet d'une autre vidéo. Ainsi, notre approche cherche à repérer une cohérence globale entre objets détectés au cours du temps sans chercher à réidentifier précisément les objets. Aucune réidentification d'objets par paire n'est effectuée : plusieurs objets différents de R peuvent trouver un même correspondant dans R'.

Notons $C(S_R, S_{R'})$ la proportion d'objets apparaissant dans R qui ont trouvé un cor-

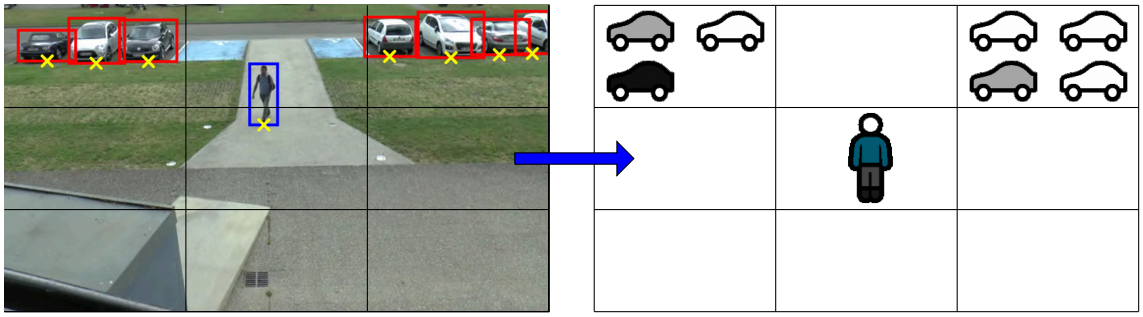


FIGURE 2.19 – Établissement des histoires – À gauche : le cadre vidéo est découpé en 9 régions et on y détecte les objets présents et leurs catégories (les rectangles bleus et rouges correspondent aux catégories respectivement « humain » et « voiture »). Le point de contact au sol est marqué d'une croix jaune. À droite : l'histoire de chaque région est constituée de la liste des objets qui y apparaissent au niveau de leur point de contact au sol et sont caractérisés par une catégorie et un descripteur d'apparence.

respondant dans $R^{'}$ et $|S_R|$ (respectivement $|S'_R|$) le nombre d'objets apparaissant dans l'histoire S_R (respectivement S'_R). Nous définissons la distance $d(S_R, S_{R'})$ entre deux histoires par :

$$d(S_R, S_{R'}) = 1 - \frac{C(S_R, S_{R'}) + C(S_{R'}, S_R)}{|S_R| + |S_{R'}|}. \tag{2.2}$$

Cette distance peut s'interpréter comme le pourcentage d'objets issus des deux histoires qui ne trouvent pas de correspondant. Pour éviter de compter une multitude de fois les objets statiques de la scène, tels que les voitures garées, nous mesurons l'indice de Jaccard, rappelé en annexe B, entre toutes les paires d'objets de même catégorie aux temps τ_i et τ_{i+1}. Les paires de détections présentant un indice supérieur à 0.9 ne sont pas prises en compte dans le calcul des histoires.

Nous allons maintenant détailler le dernier aspect de notre approche qui concerne l'établissement du graphe de liens entre vidéos à partir des distances entre les histoires des régions qui les composent à différentes résolutions.

2.6.3 Recherche multirésolution de liens entre vidéos

Dans cette section, l'algorithme de recherche multirésolution de recouvrement entre vidéos est détaillé (voir algorithme 1). Pour établir que deux vidéos présentent du recouvrement, nous calculons les distances entre leurs histoires à différentes résolutions, de la résolution la plus grossière jusqu'à la résolution la plus fine, si nécessaire.

Pour chaque paire de vidéos, les différentes résolutions contribuent au résultat comme suit. Les deux vidéos sont d'abord comparées à la résolution la plus grossière $s = 1$ en prenant en compte tous les objets qui apparaissent sur de larges intervalles de temps. Si la distance entre leurs histoires dépasse un seuil σ^s_{rejet}, dépendant du niveau de résolution

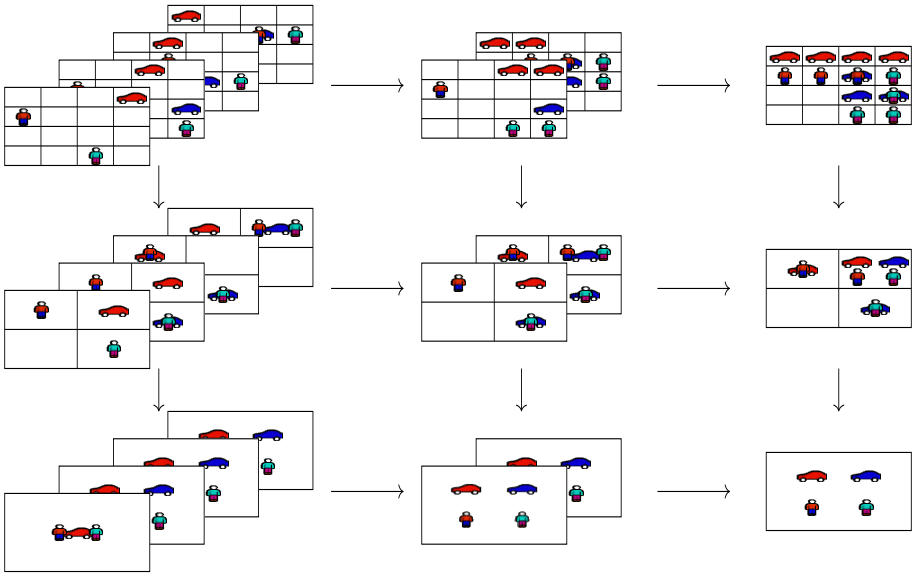


FIGURE 2.20 – Histoires multirésolutions – Un exemple d'histoire à différentes résolutions d'espace et de temps. De gauche à droite : la résolution temporelle est réduite en prenant l'union des histoires successives deux à deux. De haut en bas : la résolution spatiale est réduite en prenant l'union de cellules adjacentes. L'histoire en haut à gauche est ainsi la plus détaillée tandis que l'histoire en bas à droite est un résumé de tous les objets qui apparaissent dans la vidéo, peu importe leur emplacement ou l'instant où ils apparaissent.

s, les vidéos sont considérées comme n'ayant rien à voir et ne présentent donc pas de recouvrement. Elles ne sont pas comparées à une résolution plus fine dans ce cas. Si, au contraire, la distance entre histoires est inférieure à un seuil σ^s_{accept}, les vidéos présentent des objets similaires au cours du temps et il n'est alors pas nécessaire de chercher des correspondances à une résolution plus fine : le recouvrement est établi. Le plus souvent, la distance entre les histoires S_R et S'_R tombe entre les deux seuils. Nous subdivisons alors les régions en cours de comparaison et toutes les paires de sous-régions sont étudiées à leur tour. Les seuils σ^s_{accept} et σ^s_{rejet} s'adaptent en fonction de la résolution selon les relations suivantes :

$$\sigma^s_{accept} = 1 - (1 - \sigma^1_{accept})^s, \tag{2.3}$$

$$\sigma^s_{rejet} = (1 + \sigma^s_{accept})/2. \tag{2.4}$$

Ainsi, à mesure que les comparaisons deviennent plus fines, nous nous montrons plus sévères envers les paires de cellules qui ne se correspondent pas pour éviter de continuer à subdiviser des paires de cellules peu prometteuses. À l'inverse, nous nous montrons plus cléments envers les paires qui se correspondent car, à mesure que le voisinage de recherche autorisé se réduit, moins d'objets parviennent à trouver un correspondant. Le processus est

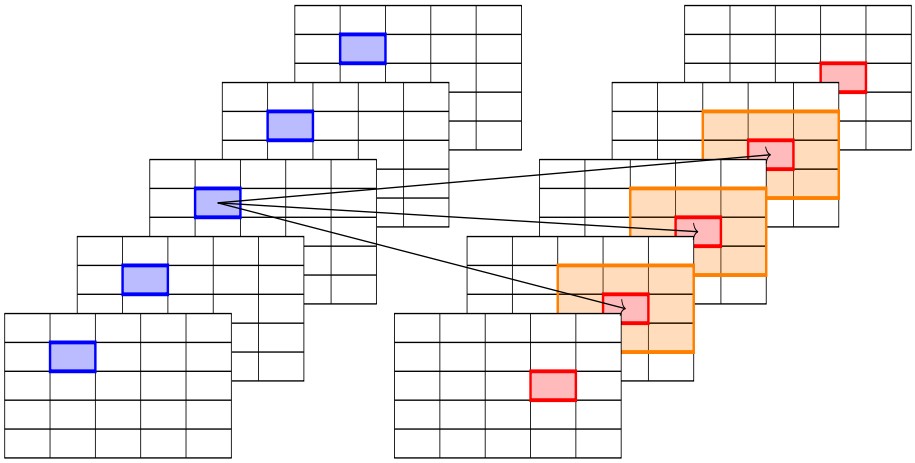
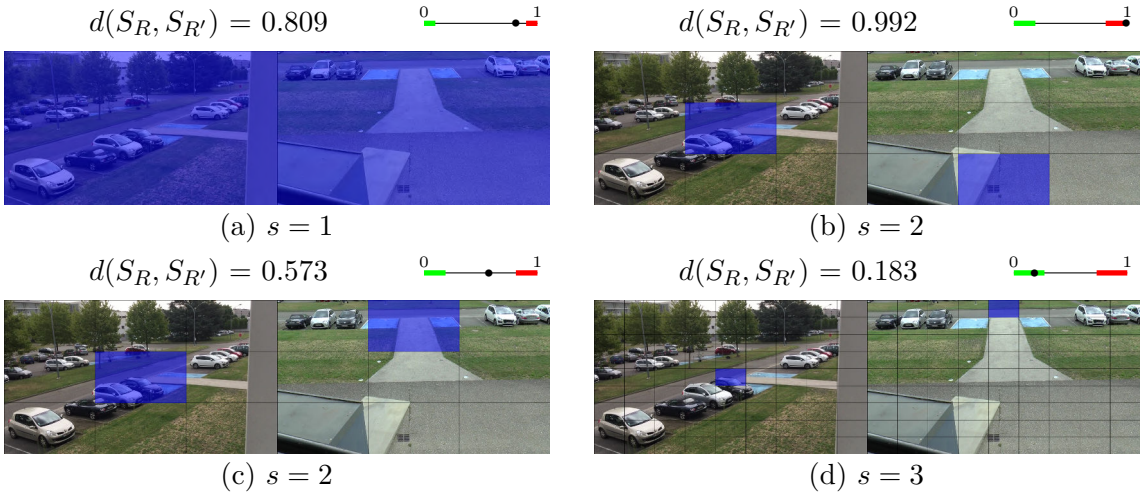


FIGURE 2.21 – Illustration du calcul de dissemblance entre deux histoires S_R (en bleu) et $S_{R'}$ (en rouge) – Pour chaque objet apparaissant dans S_R^i, on cherche un correspondant de même catégorie et d'apparence similaire dans le voisinage spatio-temporel de $S_{R'}^i$ (les régions en orange). La distance $(S_R, S_{R'})$ est alors le pourcentage d'objets de S_R et $S_{R'}$ qui ne trouvent pas de correspondant.

répété jusqu'à ce qu'il ne reste plus aucune paire de régions à comparer. Nous disposons alors d'une liste de liens entre régions de la paire de vidéos. La figure 2.22 illustre un exemple de l'application de cet algorithme de recherche de liens à plusieurs résolutions.

2.7 Conclusion

Dans ce chapitre, nous avons passé en revue les principaux outils permettant de décrire des vidéos en termes d'objets et de décor ainsi que les approches de détermination de liens entre caméras avec et sans recouvrement des champs de vue. Parmi ces approches, nous nous sommes inspirés de l'article [Loy 09] qui décrit le profil d'activité d'objets au cours du temps et du principe d'exclusion [Van Den Hengel 06] pour proposer une première contribution cherchant à décrire les vidéos à partir des objets qui y apparaissent au cours du temps. Cette première contribution a été acceptée pour publication à la conférence ICPR 2020 [Malon 20b]. Dans le chapitre suivant, nous présentons les jeux de données utilisés pour valider notre approche et les résultats obtenus.



(a) $s = 1$ (b) $s = 2$ (c) $s = 2$ (d) $s = 3$

FIGURE 2.22 – Estimation du recouvrement entre deux vidéos – En vert : zone d'acceptation. En rouge : zone de rejet. (a) Les histoires sont d'abord comparées à une échelle globale. Leur distance ne tombant dans aucune des deux zones d'arrêt, la fenêtre est subdivisée et on compare toutes les paires de sous-régions. (b) En bleu, une paire de sous-régions de (a) dont la distance entre histoires dépasse le seuil σ^2_{reject} : on ne comparera pas leurs sous-régions. (c) Une autre paire de sous-régions dont la distance entre histoires tombe entre les deux zones d'arrêt. Toutes les paires de sous-régions sont comparées. (d) Une paire de sous-régions de (c) dont la distance entre histoires est inférieure au seuil σ^3_{accept}. Ces sous-régions sont considérées comme se correspondant.

Algorithme 1 : Estimation multirésolution du recouvrement entre deux vidéos – La variable `correspondances` est l'ensemble des paires de régions qui se correspondent. La variable `candidats` est l'ensemble des triplets composés de deux régions à comparer et d'un niveau de résolution. Lorsqu'un candidat (R_1, R_2, `s`) est rejeté (instruction `Ne rien faire`), aucune correspondance n'est cherchée à une résolution plus fine. L'opérateur `Subdiv` renvoie l'ensemble des sous-régions à un niveau d'échelle plus fine.

Résultat : `correspondances` : liste des liens entre V_1 et V_2

```
correspondances = [ ]
candidats = [ (V1, V2, 1) ]
tant que candidats ≠ ∅ faire
    si d(S_R1, S_R2) ⩽ σ^s_accept alors
        correspondances = correspondances ∪ (R1, R2)
    sinon si d(S_R1, S_R2) ⩽ σ^s_rejet alors
        ∀ (r1, r2) ∈ Subdiv(R1) × Subdiv(R2)
          candidats = candidats ∪ (r1, r2, s+1)
    sinon
        Ne rien faire
    fin
    Supprimer candidats[0]
fin
Retourner correspondances
```

Chapitre 3

Expérimentations sur la recherche multirésolution de liens

Contenu

3.1 Introduction

Dans ce chapitre, nous présentons les expérimentations réalisées sur notre approche de recherche multirésolution de liens de recouvrement présentée au chapitre précédent. Nous commençons pour cela par passer en revue les différents jeux de données multicaméras publics et nous comparons leurs propriétés pour identifier ceux qui correspondent à notre problématique. Pour disposer de davantage de vidéos présentant des champs de vue en recouvrement, nous présentons également un jeu de données que nous avons créé, annoté et publié.

3.2 Jeux de données

L'un des premiers problèmes qui s'est posé est celui des données de travail. Dans le contexte de nos recherches, un jeu de données pertinent devrait contenir des vidéos présentant des champs de vue en recouvrement sur une période de temps commune. Afin qu'il soit possible d'y appliquer des approches de recherche de liens entre vidéos basées sur des détections simultanées d'objets, un certain nombre d'objets doivent apparaître dans ces vidéos. Idéalement, ces vidéos doivent correspondre à un contexte de surveillance : les déplacements et actions des personnes qui y apparaissent doivent être réalistes, c'est-à-dire agir de manière naturelle. Les approches évaluées nécessitent des vidéos avec recouvrement sur une période de temps commune suffisamment longue pour accumuler des preuves de corrélation entre les objets qui apparaissent. Dans cette partie, nous commençons par présenter succinctement plusieurs jeux de données multicaméras mais inadaptés à notre problématique. Nous évoquons ensuite plus en détails ceux qui remplissent nos exigences. Nous récapitulons les propriétés des différents jeux de données envisagés dans un tableau. Enfin, nous présentons le jeu de données ToCaDa [Malon 18], que nous avons créé, annoté et publié, et qui a été constitué à partir de nombreuses caméras présentant du recouvrement.

3.2.1 Jeux de données multivues pour la reconnaissance d'action

De nombreux jeux de données multivues ont pour thématique la reconnaissance d'action. Dans ce but, ils contiennent un ensemble de séquences vidéos où chacune montre un acteur réalisant une action individuelle en étant filmé par des caméras disposées selon différents points de vue. Parmi ces jeux de données, on retrouve :

— *Inria Xmas Motion Acquisition Sequences*, (IXMAS) [Weinland 06] dans lequel chaque acteur réalise chaque action à trois reprises ;

— *Multiview Action 3D* [Wang 14] où les séquences comportent un arrière-plan offrant un contexte et où des objets peuvent être impliqués dans certaines actions ;

— *Multicamera Human Action Video Dataset* MuHavi [Singh 10] où des données de silhouettes annotées manuellement sont fournies ;

— *Berkeley Multimodal Human Action Database* (MHAD) [Ofli 13] qui utilise conjointement des caméras classiques et des capteurs de profondeur Kinect ;

— *Nanyang Technological University RGB+D* (NTU RGB+D) [Shahroudy 16] qui utilise également des caméras classiques et des capteurs de profondeur Kinect et comporte de très nombreuses séquences (56 880).

Les nombres de caméras, d'acteurs et de types d'actions sont indiqués dans le tableau 3.1. La figure 3.1 présente des exemples d'actions et de vues provenant de certains de ces jeux de données.

Bien qu'ils disposent de points de vue multiples d'une même scène, ces jeux de données ont peu d'intérêt pour nos travaux en raison de la courte durée des vidéos, du manque de réalisme et du fait qu'une seule personne est présente dans chaque vidéo. De plus, nous n'employons pas d'approche de reconnaissance d'action dans nos contributions. Nous ne les utilisons donc pas dans nos expérimentations.

Jeu de données	Nombre de points de vue	Nombre d'actions	Nombre d'acteurs	Spécificités
IXMAS [Weinland 06]	5	11	10	3 séquences par action
Multiview Action 3D [Wang 14]	3	10	10	Actions impliquant des objets
MuHaVi [Singh 10]	8	17	14	Annotations des silhouettes
MHAD [Ofli 13]	4	11	12	RGB + Kinect, 5 séquences par action
NTU RGB+D [Shahroudy 16]	3	60	40	RGB + Kinect, très nombreuses séquences

Tableau 3.1 – Tableau récapitulatif des principaux jeux de données multivues pour la reconnaissance d'actions – Lorsque le nombre de séquences par action n'est pas indiqué, chaque action n'est réalisée qu'à une seule reprise.

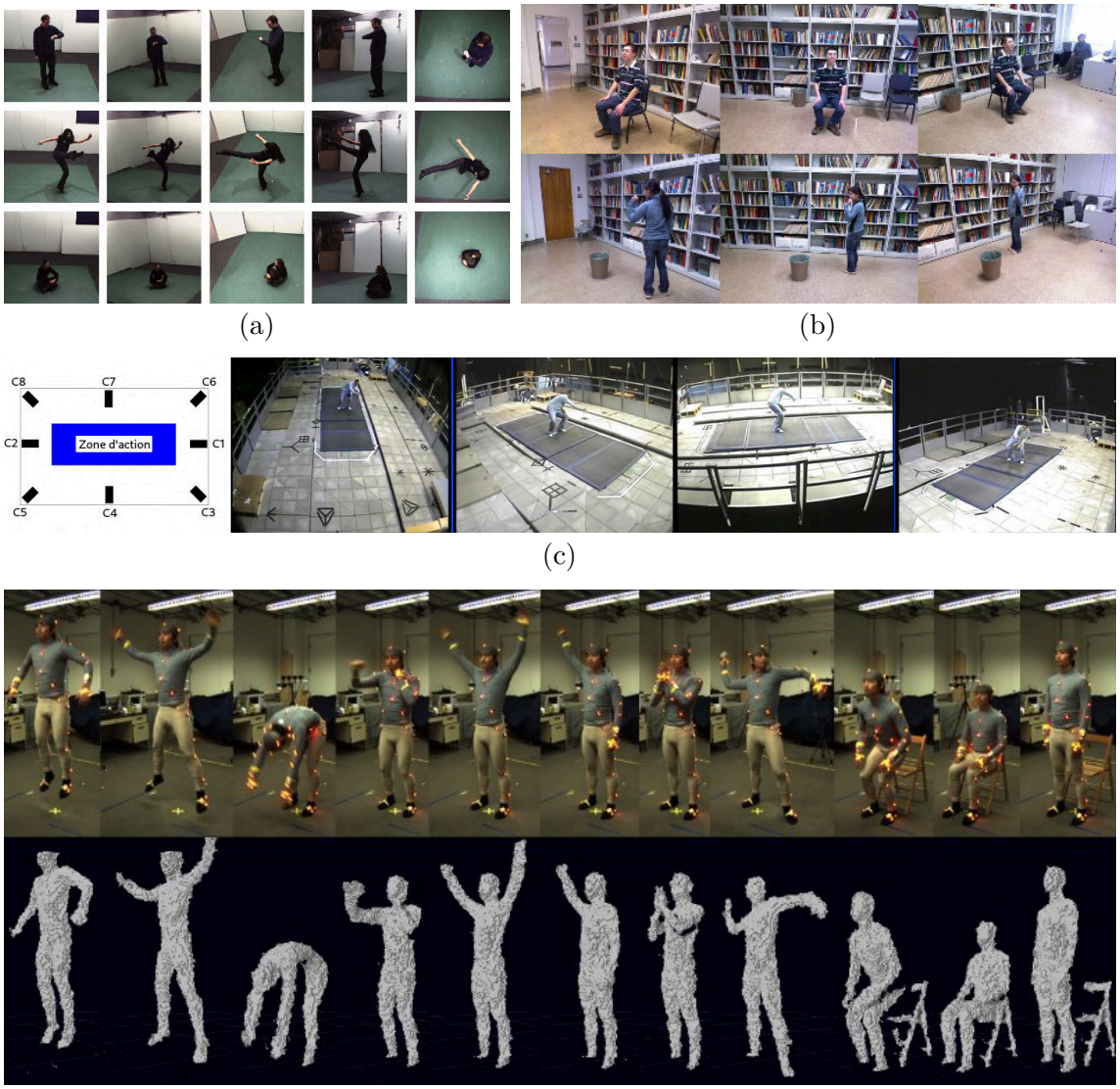


FIGURE 3.1 – Quelques vues des jeux de données multicaméras pour la reconnaissance d'actions – (a) IXMAS [Weinland 06] : les lignes correspondent respectivement aux actions « regarder sa montre », « donner un coup de pied » et « s'asseoir », et les colonnes correspondent aux différentes caméras. (b) *Multiview Action 3D* [Wang 14] : les lignes correspondent respectivement aux actions « s'asseoir » et « jeter à la poubelle » et les colonnes correspondent aux différentes caméras. (c) MuHaVi [Singh 10] : de gauche à droite, nous montrons la disposition des 8 caméras autour de la scène et les images issues de 4 des caméras (dans l'ordre 1, 3, 4 et 6) pour l'action « dessiner un graffiti ». (d) MHAD [Ofli 13] : clichés des différentes actions disponibles dans le MHAD et nuages de points correspondants obtenus à partir des données de profondeur des périphériques Kinect. De gauche à droite, les actions sont : « sauter », « sauter avec écart latéral », « se pencher », « donner un coup de poing », « agiter les deux mains », « agiter une seule main », « taper dans ses mains », « lancer », « s'asseoir », « être assis » et « se tenir debout ».

3.2.2 Jeux de données multivues pour la vidéosurveillance

Plusieurs jeux de données spécifiques à la vidéosurveillance ont été proposés par la communauté scientifique. Les applications ciblées sont généralement l'analyse de scène, la réidentification et l'estimation de la topologie d'un réseau de caméras. Dans cette partie, nous listons les principaux jeux de données envisagés en apportant une attention particulière aux propriétés listées ci-dessous :

1. Le **nombre total de caméras**. Disposer d'un maximum de caméras nous permet d'évaluer la capacité de nos approches à gérer des jeux de données de grande taille.
2. Le **nombre de caméras avec des champs de vue en recouvrement**. Il est important que nous en disposions en nombre significatif car cela correspond au contexte de nos recherches.
3. Le **nombre d'entités différentes apparaissant dans les vidéos et la durée des vidéos**. Pour évaluer notre méthode de recherche de liens par comparaison d'histoire qui s'appuie sur la cohérence globale des détections simultanées dans le temps, nous avons besoin de vidéos dans lesquelles apparaissent différents nombres d'entités et sur des durées différentes.
4. Les **nombres moyen et maximum d'entités apparaissant simultanément**. Ce sont des indicateurs qui reflètent la complexité d'un jeu de données.
5. La **synchronisation temporelle des caméras**. Elle est essentielle pour évaluer notre approche de recherche de liens de recouvrement car c'est une des hypothèses que l'on a faites.
6. Les **catégories d'objets en présence**. Dans un scénario de vidéosurveillance, on retrouve généralement des piétons ou des véhicules tels que des voitures, des camions, des motos et des vélos. Certains jeux de données comportent plusieurs catégories d'objets différents, d'autres sont spécifiques à un type d'objets.
7. Le **type d'environnement (intérieur/extérieur)**. Les scènes d'extérieur sont généralement plus étendues et sujettes à contenir des objets espacés les uns des autres tandis que les scènes d'intérieur contiennent généralement un nombre limité de personnes peu espacées. Il sera intéressant d'évaluer l'impact de ces deux types d'environnement sur les résultats. Nous relevons donc, pour chaque jeu de données, s'il contient des scènes d'intérieur, d'extérieur ou les deux.

Le tableau 3.2 situé page 53 récapitule ces différentes propriétés pour les différents jeux de données présentés.

Context Aware Vision using Image-based Active Recognition (CAVIAR)

Dans le cadre du projet CAVIAR (http://homepages.inf.ed.ac.uk/rbf/CAVIAR/), un certain nombre de vidéos ont été enregistrées, mettant en scène plusieurs scénarios

d'intérêt. Il s'agit de personnes marchant seules, rencontrant d'autres personnes, regardant des vitrines, entrant et sortant de magasins, se battant, s'évanouissant ou encore laissant tomber un paquet dans un lieu public. Toutes les séquences ont été capturées à l'aide de caméras grand angle et sont de faible résolution (384 × 288 pixels). Une partie de ces séquences a été filmée par une seule caméra fixée dans l'entrée du laboratoire de l'INRIA à Grenoble. Le reste des séquences provient d'une paire de caméras placées dans une allée d'un centre commercial à Lisbonne. L'une est située face à l'allée tandis que l'autre est placée perpendiculairement et toutes deux sont synchronisées temporellement. Ce sont ces vidéos qui correspondent à notre cas d'usage. Leurs durées varient d'une dizaine de secondes à une minute selon la séquence. Pour ce jeu de données, nous indiquons la durée cumulée de toutes les séquences dans le tableau 3.2. Les données de référence comportent les boîtes englobantes des différentes personnes et objets apparaissant dans les vidéos, ainsi que des mesures de distance entre certains points permettant d'estimer la transformation homographique entre les images du plan du sol. Des images de ces vues, ainsi que les points fournis pour le calcul de l'homographie sont montrés sur la figure 3.2. Des explications sur les homographies sont données en annexe B.

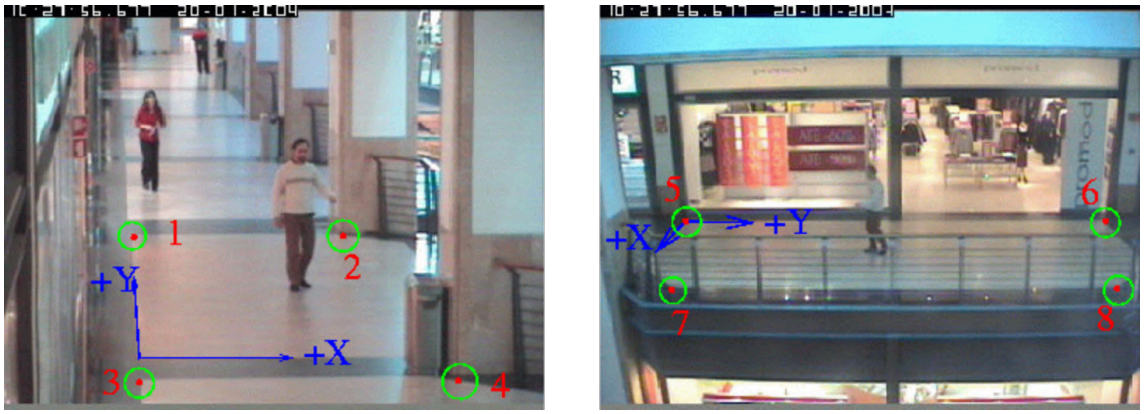


FIGURE 3.2 – Exemples de vues du jeu de données CAVIAR — Clichés des deux vues de CAVIAR provenant d'un centre commercial. Les auteurs indiquent, pour chaque point, ses coordonnées pixelliques ainsi que ses coordonnées en cm dans la scène selon les axes X et Y. Les correspondances entre les paires de points (1, 6), (2, 8), (4, 7) et (3, 5) permettent alors de calculer la transformation homographique qui relie n'importe quel point du plan dans l'image de gauche à son correspondant dans l'image de droite. Images issues de `https://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/`.

Video and Image Retrieval and Analysis Tool (VIRAT)

Le jeu de données VIRAT [Oh 11] contient un grand nombre de vidéos issues de caméras de surveillance en haute résolution (1920 × 1080). Dans ce jeu de données, 16 scènes ont été enregistrées pendant de nombreuses heures, puis seules les 25 heures de vidéos présentant le plus d'activité ont été gardées. Des vidéos aériennes sont également dispo-

nibles. Seules deux paires de caméras présentent du recouvrement dans leurs champs de vue. Toutefois, ce recouvrement représente une très faible proportion de l'image. Chaque objet mobile de la scène a été annoté manuellement par des boîtes englobantes tout le long de son déplacement. Les annotations contiennent aussi trois catégories d'évènements : les interactions entre personnes (par exemple serrer la main), les interactions entre une personne et un véhicule (par exemple ouvrir la portière) et les interactions entre une personne et un bâtiment (par exemple entrer dans un bâtiment), pour un total de 23 types d'évènements. En dehors de quelques acteurs dont le comportement est scénarisé, la majorité des personnes à l'écran ne font pas partie du scénario et agissent naturellement, rendant les vidéos particulièrement réalistes. La figure 3.3 montre quelques unes des vues disponibles dans le jeu de données.

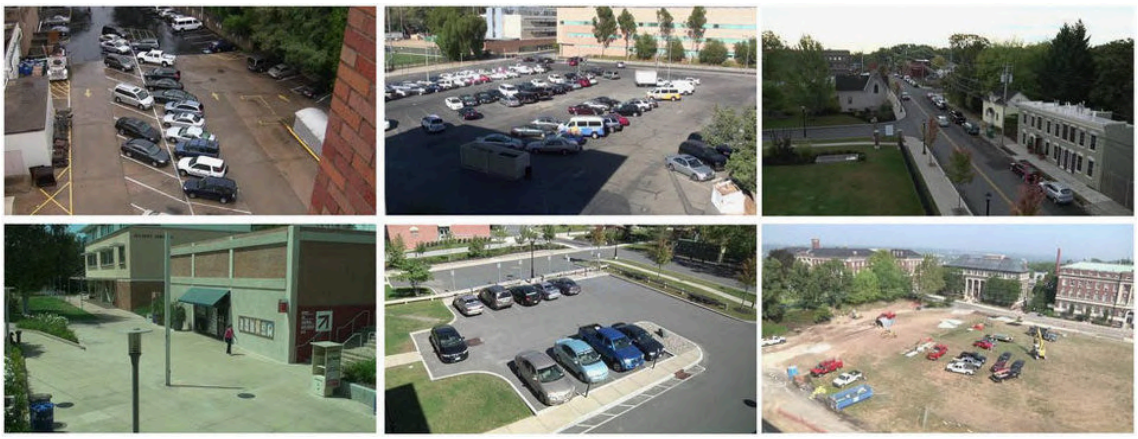


FIGURE 3.3 – Exemple de séquences du jeu de données VIRAT [Oh 11] — À l'exception de quelques caméras placées dans l'allée d'un campus universitaire, les autres filment des parkings. Seules deux caméras présentent un recouvrement, assez limité, dans leurs champs de vue. Toutes les autres ont des champs de vue disjoints. Images issues de [Vishwakarma 13].

Multi-camera Pedestrian Videos

L'EPFL a publié en 2007 un jeu de données multivues [Fleuret 07] intitulé *Multi-camera Pedestrian Videos* et composé de 5 scénarios différents, avec, pour chaque scénario, 3 à 4 caméras filmant la même scène de façon synchronisée. Les vues pour chaque scénario sont affichées sur la figure 3.4. Selon le scénario, le nombre maximum de personnes présentes simultanément dans la scène varie de 3 à plus d'une dizaine pour le match de basket. La durée des scénarios va de 2 minutes pour le plus court à 6 minutes pour le plus long — à nouveau le match de basket. Les vidéos sont de faible résolution (360×288). Dans leur article [Fleuret 07], les auteurs utilisent ce jeu de données pour calculer les positions des différents personnages apparaissant dans la scène sur une vue de haut à partir de leurs positions détectées dans chacune des vues.

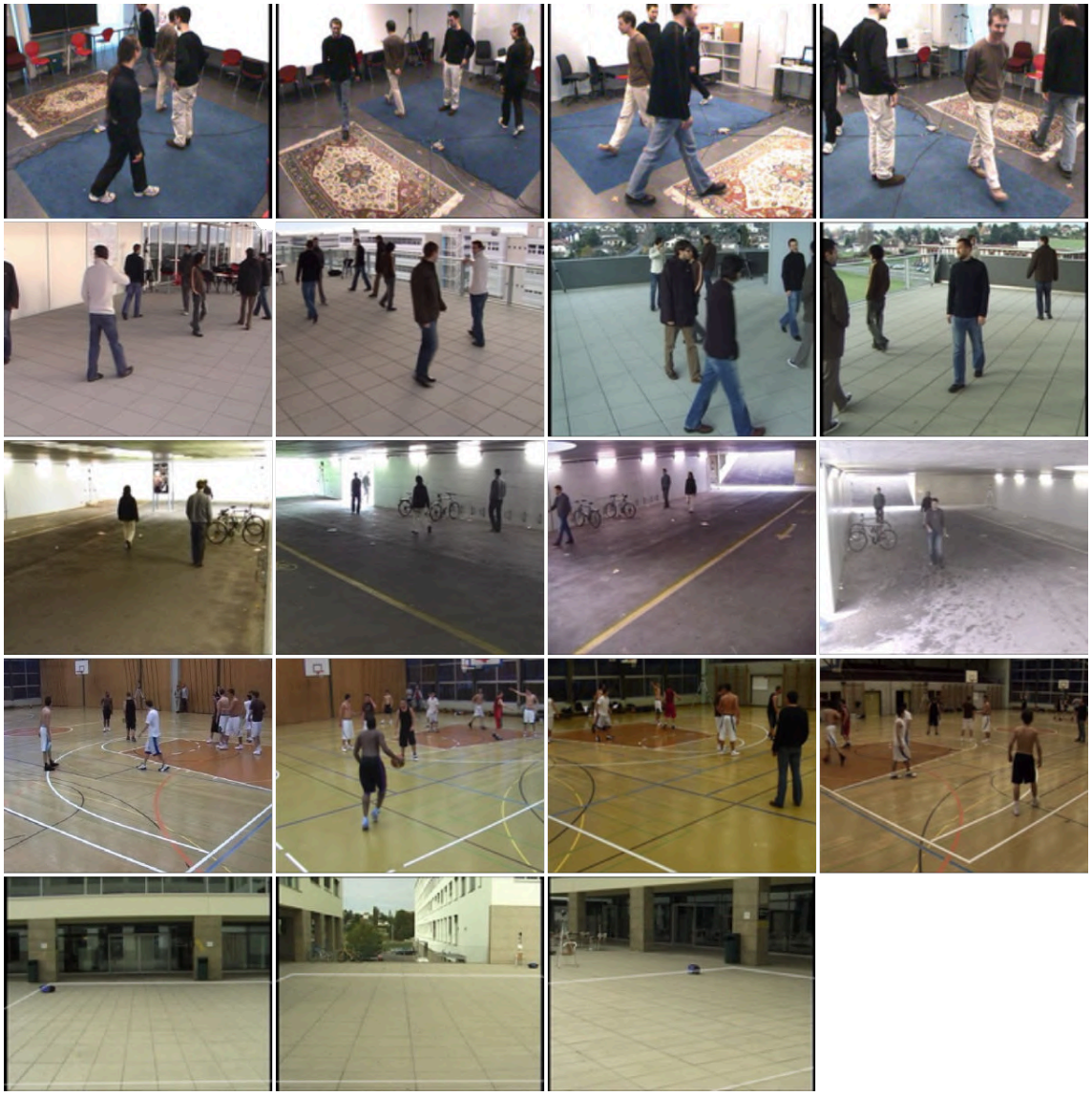


FIGURE 3.4 – Différentes vues du jeu de données EPFL [Fleuret 07] — Par ligne, les scénarios sont dans l'ordre : salle, terrasse, passage, basket et campus. Images extraites de https://www.epfl.ch/labs/cvlab/data/data-pom-index-php/.

WILDTRACK

L'EPFL a proposé un autre jeu de données multivues, WILDTRACK [Chavdarova 18]. Ce jeu de données présente un défi car il y a beaucoup de circulation sur la place filmée par les 7 caméras. Pour chaque caméra, on dispose d'une vidéo de 35 minutes en haute résolution (1920 × 1080). De plus, les piétons se déplacent généralement en groupes : on se trouve dans une situation de foule où il est difficile de discerner les différents individus. Ce jeu de données est illustré par la figure 3.5.

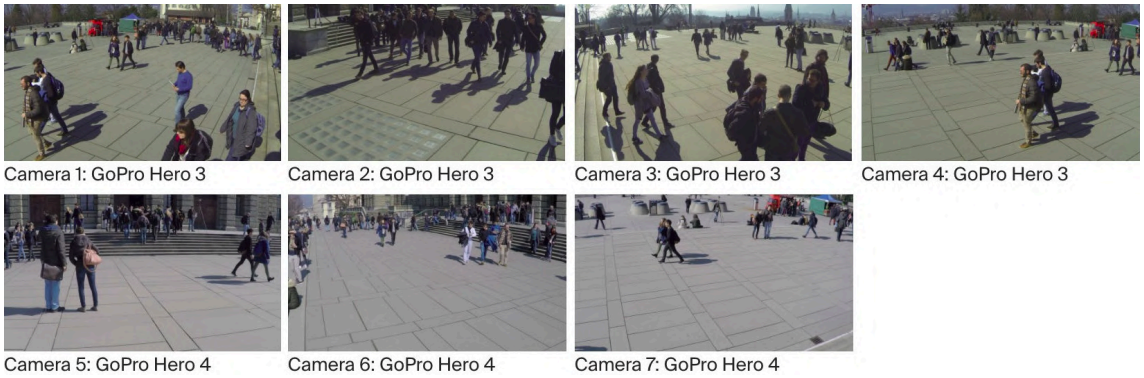


FIGURE 3.5 – Jeu de données WILDTRACK [Chavdarova 18] — Les 7 points de vue et les types de caméras utilisés sont affichés. Images issues de https://www.epfl.ch/labs/cvlab/data/data-wildtrack/.

CityStreet

Le jeu de données CityStreet [Zhang 19] a été réalisé en filmant un carrefour pendant une heure avec 5 caméras dont les champs de vue se recouvrent. Les images sont de très haute résolution (2704 × 1720). De nombreux piétons y apparaissent simultanément, ainsi que des véhicules. De nouveau, il s'agit d'un jeu de données où l'on trouve des situations de foules. Après l'acquisition de ces vidéos, 500 instants ont été choisis uniformément au cours de cette heure et les images de chaque caméra à ces instants ont été extraites et constituent le jeu de données. La figure 3.6 illustre quelques unes des vues ainsi que le plan de disposition des 5 caméras.

FIGURE 3.6 – Jeu de données CityStreet [Zhang 19] — À gauche, les vues de 3 des 5 caméras du jeu de données CityStreet. À droite, la disposition des 5 caméras sur un plan vu du ciel. Images issues de http://visal.cs.cityu.edu.hk/downloads/.

PETS2009

Le jeu de données *PETS2009* [Ferryman 09] contient un ensemble de vidéos filmées depuis 8 caméras aux champs de vue en recouvrement. Ces vidéos mettent en scène trois scénarios différents où ont notamment lieu des mouvements de foules. De nombreux individus sont proches les uns des autres et se déplacent généralement en se suivant [Moussaïd 11]. Dans ce jeu de données, on trouve des situations :

— d'évacuation, où les individus formant la foule se dispersent individuellement ;

— de dispersion locale, où la majorité des individus forment une foule tandis que quelques individus s'en éloignent ;

— de division, où la foule formée d'un seul groupe d'individus se divise en plusieurs groupes disjoints ;

— de regroupement, où une foule se forme à partir d'individus isolés ou de groupes d'individus.

D'une situation à une autre, les vidéos durent entre une dizaine de secondes et une minute, et 4 à 7 des 8 caméras filment la scène. La figure 3.7 présente les vues des 8 caméras.

FIGURE 3.7 – Les 8 différentes vues du jeu de données *PETS2009* [Chavdarova 18]. Images issues de `http://www.cvg.reading.ac.uk/PETS2009/a.html`.

Multiview Extended Video with Activities (MEVA)

Lors de la conférence *Advanced Video and Signal-based Surveillance* (AVSS) de 2019, un nouveau jeu de données de grande ampleur a été présenté. Il contient des enregistrements vidéo d'un campus universitaire sur 7 jours et 6 nuits, découpés en séquences de 30 minutes pour près de 30 caméras synchronisées, dont quelques caméras infrarouges. Le site est découpé en 4 parties principales : les bâtiments scolaires, l'arrêt de bus, la station de métro et le parking. La figure 3.8 montre la disposition des différentes caméras sur le campus. Parmi les séquences actuellement disponibles, 6 paires de caméras offrent du recouvrement dans leurs champs de vue. Ce jeu de données est particulièrement adapté pour faire du suivi et de la réidentification car les différents piétons apparaissent dans de nombreuses caméras au fil de la journée. Seule une partie des annotations est actuellement disponible. La figure 3.9 montre quelques unes des nombreuses vues disponibles.

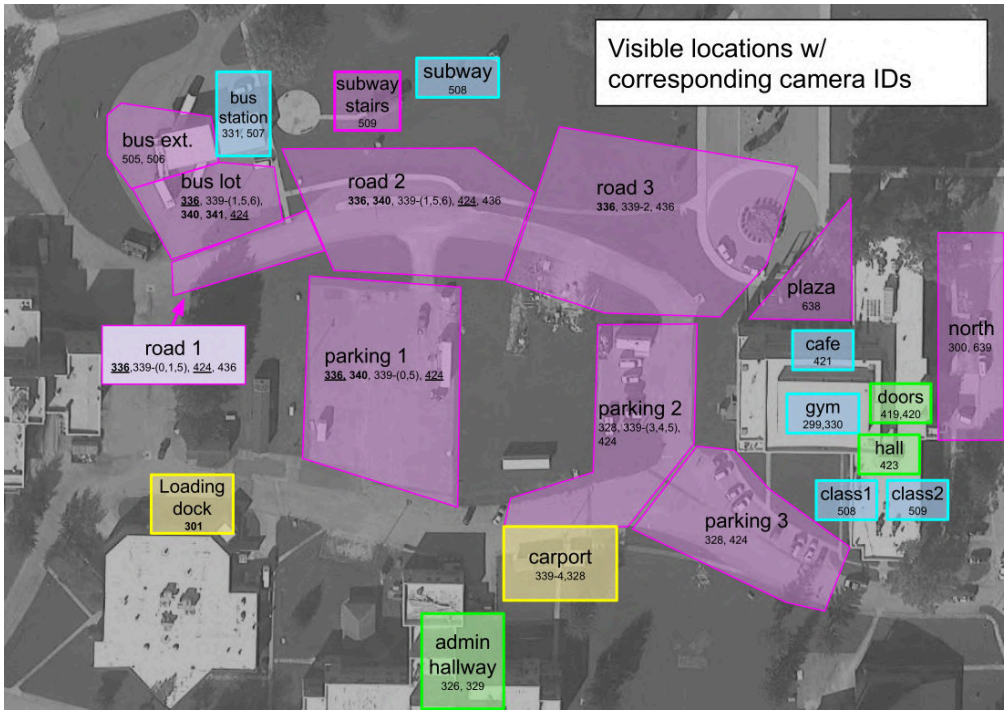


FIGURE 3.8 – Disposition des caméras du jeu de données MEVA — Les identifiants des caméras sont indiqués. Le code couleur indique, d'une part, si la caméra filme en intérieur (bleu ou vert) ou en extérieur (jaune ou rose) et, d'autre part, s'il s'agit d'un lieu de passage (en vert ou rose) ou d'un lieu où l'on s'installe (bleu ou jaune). Image issue de https://mevadata.org/.

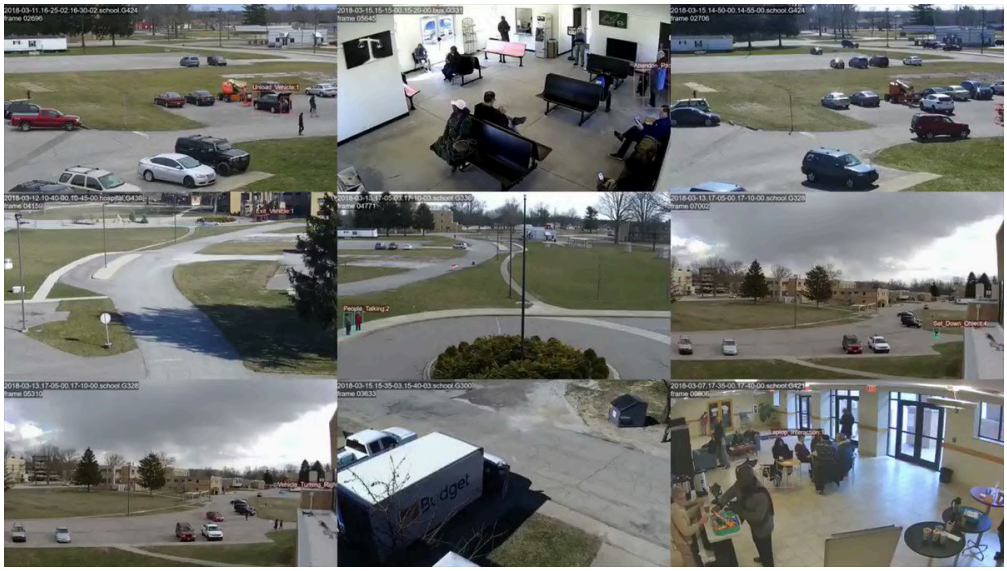


FIGURE 3.9 – Différentes vues du jeu de données MEVA — Les séquences sont très hétérogènes en termes de propriétés : en intérieur / en extérieur, lieu de transit / lieu où l'on s'installe, très fréquenté / peu fréquenté. Images issues de https://mevadata.org/.

La plupart du temps, il n'y a pas d'activité dans ces vidéos. En conséquence, pour chacune des 6 paires de caméras dont les champs de vue se recouvrent, nous avons sélectionné des séquences synchronisées de 5 minutes présentant de l'activité.

3.2.3 Jeux de données devenus inaccessibles

Au cours de nos recherches, nous avons identifié deux jeux de données très utilisés par la communauté scientifique mais qui ne sont aujourd'hui plus accessibles. Nous les présentons donc brièvement. Ils n'apparaissent pas dans le tableau récapitulatif 3.2.

Video Surveillance Online Repository (ViSOR)

La base de données ViSOR [Vezzani 08] a été conçue en tant que répertoire pour collecter, partager et annoter des données de vidéosurveillance. Conçue au départ pour être mise à jour continuellement, elle n'est aujourd'hui plus maintenue et n'est plus accessible. Son interface, présentée sur la figure 3.10 permettait de choisir des catégories de vidéos dans un catalogue en contenant plusieurs milliers.

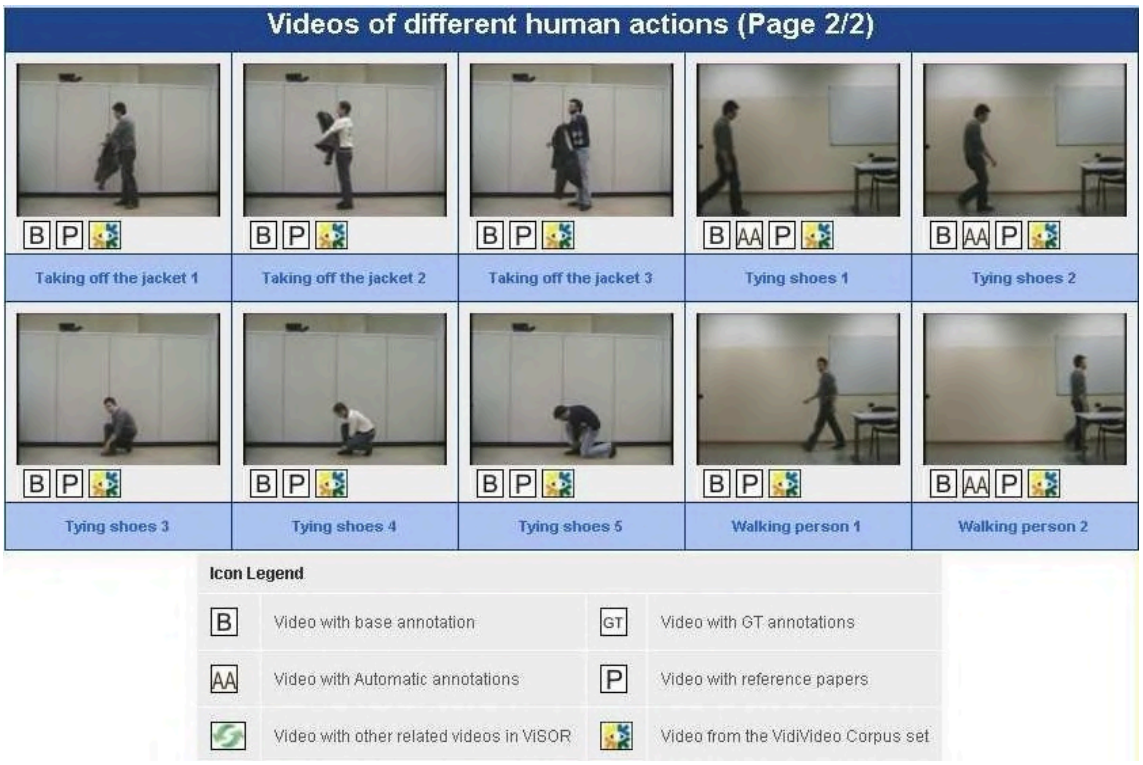


FIGURE 3.10 – Plate-forme de partage de données de vidéosurveillance ViSOR [Vezzani 10] — Pour une catégorie donnée, ici des actions humaines, l'interface affichait l'ensemble des vidéos correspondant ainsi que les fichiers d'annotations disponibles pour chaque séquence. Il était possible de contribuer à annoter des vidéos.

	Nombre de caméras	dont en recouvrement	Durée moyenne par vidéo	Synchronisation temporelle	En intérieur	En extérieur	Uniquement piétons	Uniquement véhicules	Nombre d'objets différents en moyenne	Nombre d'objets différents au maximum
CAVIAR	2	2	5m00	●		■	▲		≃50	≃40	≃50
PETS2009 scénar.1	4	4	0m30	●		■	▲		≃50	≃40	≃50
PETS2009 scénar.2	7	7	1m13	●		■	▲		≃10	≃5	≃10
PETS2009 scénar.3	6	6	0m15	●		■	▲		≃8	≃6	≃8
CityStreet	5	5	1h00	●		■			≥1000	≃50	≃80
VIRAT	16	4	1h30			■			≃70	≃5	≃10
EPFL basket	4	4	6m14	●	■		▲		≃16	≃8	≃12
EPFL campus	3	3	3m55	●		■	▲		≃3	≃1	≃3
EPFL passage	4	4	1m40	●	■		▲		≃8	≃2	≃4
EPFL salle	4	4	2m00	●	■		▲		≃6	≃3	≃6
EPFL terrasse	4	4	3m00	●		■	▲		≃7	≃4	≃7
WILDTRACK	7	7	35m00	●		■	▲		≃300	≃30	≃50
MEVA basket	32	6×2	56h00	●	●	●			≃4	≃2	≃4
MEVA bus	32	6×2	56h00	●	●	●			≃30	≃10	≃20
MEVA bâtiment	32	6×2	56h00	●	●	●			≃40	≃4	≃15
MEVA couloir	32	6×2	56h00	●	●	●			≃8	≃2	≃3
MEVA campus	32	6×2	56h00	●	●	●			≃100	≃20	≃50
MEVA station	32	6×2	56h00	●	●	●			≃12	≃3	≃6
Youtube région 1	2	1	5m00	●		■	▲		≃100	≃10	≃20
Youtube région 2	2	2	5m00	●		■			≃120	≃5	≃15
Youtube région 3	3	2	5m00	●		■			≃200	≃15	≃30
ToCaDa	25	19	5m24	●		■			≃30	≃3	≃8

Tableau 3.2 – Tableau récapitulatif des jeux de données multivues pour la vidéosurveillance – Les éléments positifs (synchronisation des vidéos ou présence à la fois de scènes d'intérieur et d'extérieur) sont mis en évidence par des points vers tandis que les éléments plutôt négatifs (présence d'une seule catégorie d'objets) sont indiqués par un triangle rouge. Les lignes en jaune correspondent à des jeux de données qui n'ont pas été publiés. La ligne en orange correspond au jeu de données que nous avons publié.

Duke MTMC

L'un des jeux de données les plus adaptés à un scénario de vidéosurveillance et parmi les plus complets en termes d'annotations et de durées des vidéos est Duke MTMC [Ristani 16]. Il contient en effet plus de 2 millions d'images de 2 834 personnes filmées sur un campus universitaire par 8 caméras pendant plus d'une heure. Deux paires de caméras présentent du recouvrement dans leurs champs de vue. En mai 2019, suite à une enquête du *Financial Times*[1] révélant l'utilisation qui était faite de ce jeu de données par certains gouvernements, les auteurs ont décidé de ne plus laisser accessible ce jeu de données.

3.2.4 Flux vidéo de caméras en direct

En recherchant les mots clés « *live camera* » sur `Youtube`, de nombreux résultats permettent de regarder en direct la vue de caméras disposées en un point donné de la planète, généralement dans des grandes villes. Nous en avons identifié 7 qui correspondent à notre cas d'usage : 2 paires disjointes présentent du recouvrement dans leurs champs de vue, 2 autres sont placées proches l'une de l'autre le long d'une allée mais ont des champs de vue qui ne se coupent pas. Enfin, l'arrière-plan de la septième vidéo est très similaire à l'arrière-plan d'une des six autres sans pour autant qu'il y ait de recouvrement dans leurs champs de vue. Vous avons téléchargé de façon synchronisée une heure de chacune de ces vidéos et avons isolé un passage de 5 minutes au cours duquel plusieurs objets apparaissent afin de pouvoir profiter des détections concomitantes nécessaires à notre approche. Nous ne pouvons pas partager les liens vers ces vidéos car elles ne sont aujourd'hui plus disponibles sur `Youtube`. Nous souhaitons néanmoins partager les résultats obtenus sur ces vidéos.

3.2.5 Un jeu de données de synthèse : Manzalab

Lorsque les données dont on dispose ne sont pas assez représentatives des cas que l'on souhaite traiter, l'utilisation de données de synthèse peut être un complément pertinent. La société Manzalab (`https://www.manzalab.com/`), partenaire du projet VICTORIA et spécialisée dans les jeux sérieux — un domaine qui consiste à emprunter les codes propres au jeu vidéo dans d'autres buts que le divertissement — a produit un outil permettant de simuler un réseau de caméras. À partir d'une scène en 3 dimensions au sein de laquelle se déplacent des personnes selon un scénario préétabli, l'outil permet de choisir la position et l'orientation de la caméra et de générer une vidéo depuis ce point de vue demandé. L'outil permet également de spécifier plusieurs paramètres :

— Le nombre de personnages apparaissant dans le scénario (de 10 à 70), chacun d'entre eux ayant des actions qui lui sont propres et qui sont déterministes. Ainsi, lorsque

1. `https://www.ft.com/content/cf19b956-60a2-11e9-b285-3acd5d43599e`

l'on augmente le nombre de personnages de 1, les personnages déjà présents ne sont pas impactés, il y a simplement un personnage supplémentaire dans la scène.

— L'heure de la journée, qui influence la clarté du ciel et de la scène de manière générale, mais n'a aucune incidence sur la position ou les actions des personnages.

— La présence ou non d'objet abandonné. Lorsque cette option est activée, l'un des personnages laisse tomber un sac au cours de son trajet.

L'outil permet de produire des vidéos d'une durée maximale de 5 minutes, le scénario n'ayant pas été écrit au-delà. La figure 3.11 présente des images issues de vidéos générées par l'outil, pour un même instant du scénario, depuis deux points de vue différents et pour trois valeurs différentes du nombre de personnes dans la scène.

FIGURE 3.11 – Exemples de vues du jeu de données Manzalab – Deux vues générées à partir du jeu de données de synthèse Manzalab avec différents nombres de personnes dans la scène.

L'utilisation de cet outil nous permet de disposer d'autant de vidéos d'une même scène que voulues. La possibilité de régler le nombre de personnages présents est intéressante pour évaluer l'impact de ce paramètre sur les méthodes d'analyse de scène. Enfin, l'outil fournit également les données de référence pour chaque vidéo générée, à savoir les coordonnées des boîtes englobantes correspondant à chaque personnage au cours du temps. Toutefois, plusieurs problèmes se posent :

— l'impossibilité de générer des vidéos d'une durée supérieure à 5 minutes, qui limite les possibilités d'étudier l'impact de la quantité de données disponibles dans l'analyse de la scène ;

— le manque de réalisme du rendu, notamment la teinte globale de la scène, très claire, mais surtout les personnages qui se ressemblent tous : ils font la même taille, ils ont des vêtements de couleur unie et leurs visages ne sont pas texturés ;

— l'interface de génération des vidéos permet d'entrer les coordonnées de la caméra et

son orientation, mais il n'est pas possible de prévisualiser la vidéo avant la fin du rendu, ce qui rend le choix du point de vue d'une caméra difficile.

Ainsi, bien que nous disposions de cet outil, nous ne l'utiliserons pas dans les expérimentations présentées dans ce mémoire. Nous trouvons toutefois pertinent d'évoquer l'existence de ce type de jeux de données et nous prévoyons de nous en servir pour des travaux à venir.

Parmi les différents jeux de données multivues présentés, tous ne sont pas exploitables par notre approche. Nous excluons notamment toutes les vidéos d'une durée inférieure à une minute. D'autre part, une contrainte matérielle limite les expériences que nous pouvons mener. En effet, pour chaque vidéo, nous stockons les coordonnées des boîtes englobantes correspondant aux objets détectés, ainsi que plusieurs descripteurs pour chaque boîte englobante. Ces traitements demandent un espace de stockage important et demandent des temps de calcul non négligeables. Pour toutes ces raisons, nous avons sélectionné spécifiquement les jeux de données qui correspondent le mieux à notre problématique : il s'agit des jeux de données *Multi-camera Pedestrian Videos* de l'EPFL (que nous nommerons simplement EPFL), MEVA, des vidéos de caméras en direct téléchargées de `Youtube`, ainsi que de notre jeu de données ToCaDa que nous allons présenter dans la section suivante.

3.3 Création et annotation d'un nouveau jeu de données multivues

Nous avons spécifiquement présenté des jeux de données multivues car nos contributions visent à trouver des liens de recouvrement entre vidéos. Par la suite, nous présenterons également dans le chapitre 5 une approche de classement des vidéos par ordre de pertinence depuis une vidéo courante. Nous avons donc besoin d'un grand nombre de vidéos présentant conjointement du recouvrement dans leurs champs de vue. Peu des jeux de données multivues identifiés contiennent plus de 5 caméras avec recouvrement.

Nous avons donc entrepris de créer, d'annoter et de publier un nouveau jeu de données avec l'aide de deux autres équipes de l'IRIT, SAMOVA et SIG, impliquées elles aussi sur des thématiques d'enquêtes. Du nom de *Toulouse surveillance Campus Dataset* (ToCaDa), il a été conçu pour convenir aux différentes équipes.

3.3.1 Création du jeu de données

En réunissant de nombreux collègues sur le campus de l'université Paul Sabatier de Toulouse, nous avons constitué un large réseau de caméras fixes (tant par leurs positions que par leurs orientations), disposées à différents endroits du campus. Deux scénarios ont été préalablement écrits et mettent en scène des allers et venues de plusieurs personnes

et véhicules dans l'ensemble du campus. Les rôles d'acteurs ont été attribués à certains, tandis que les autres avaient pour rôle de filmer la scène depuis un point de vue assigné. La plupart des actions réalisées par les personnages sont banales, mais quelques actions suspectes ont été ajoutées, à savoir un stationnement gênant, et un homme à l'attitude suspecte, qui regarde sans cesse autour de lui, et transporte un grand carton.

Au total, ce sont 25 dispositifs d'acquisition (caméras et smartphones), placés en différents endroits du campus, qui ont enregistré simultanément des séquences vidéos couvrant le même intervalle de temps. En dehors des équipes REVA et MINDS dont je fais partie, plusieurs équipes de recherche de l'IRIT travaillent sur des thématiques d'enquêtes. Nous avons donc produit un jeu de données qui satisfasse les contraintes de chaque équipe :

— Certaines vidéos présentent du recouvrement dans leurs champs de vue, en vue d'utiliser des méthodes de stéréovision au sein des équipes REVA et MINDS. 17 caméras sont disposées autour d'un même bâtiment. 2 autres caméras filment la rue devant ce bâtiment dans toute sa longueur depuis les deux extrémités. Enfin, les 6 autres caméras ont des vues disjointes et sont dispersées dans le campus.

— L'audio, domaine de recherche de l'équipe SAMOVA, a été enregistré et conservé avec les fichiers vidéos. Deux capteurs supplémentaires, uniquement sonores, ont également été disposés dans la scène,

— Dans le cadre des travaux de l'équipe SIG, les coordonnées GPS de certaines caméras ont été relevées, et certains personnages apparaissent dans plusieurs caméras différentes au cours de chaque scénario.

Les dispositifs d'acquisition étant très variés, il n'a pas été possible de récupérer directement l'heure d'enregistrement de chaque appareil. Ainsi, afin de pouvoir synchroniser précisément les vidéos les unes par rapports aux autres, un signal sonore, audible sur tout le campus, a été émis à trois reprises au moyen d'une corne de brume. Au premier signal, tout le monde commence à enregistrer, de sorte que le deuxième signal, 10 secondes plus tard, soit capturé par tous les appareils. C'est à partir de ce deuxième signal que les vidéos ont été synchronisées. Enfin, le troisième signal marque la fin du scénario. Une fois toutes les vidéos récupérées, j'ai utilisé l'outil de montage `kdenlive` [2] — qui permet notamment de découper une vidéo en sections aussi courtes que l'on veut (jusqu'à des blocs d'une seule image) — pour déterminer précisément l'image correspondant au temps de la vidéo où le second signal commence à être audible. Le son se propageant à près de 340 mètres par seconde dans l'air, et les caméras les plus éloignées étant situées à près de 200 mètres l'une de l'autre, on peut estimer un décalage maximal de l'ordre d'une demi-seconde. Néanmoins, dans le cadre de cette thèse, ce sont avant tout les caméras dont les champs de vue s'intersectent qui nous intéressent. Ces caméras-là étant très proches, on peut estimer que le décalage maximal est inférieur à un dixième de seconde.

2. https://kdenlive.org/fr/

3.3.2 Annotation des vidéos

Une fois les vidéos synchronisées, nous les avons annotées pour pouvoir évaluer différents types d'approches. Afin de rendre notre jeu de données utilisable dans les domaines couverts par les différentes équipes, nous avons produit les annotations suivantes pour chaque vidéo :

— Des annotations vidéos pour la détection, le suivi et la réidentification d'objets inter-caméra. Plus précisément, pour chaque objet présent dans la vidéo, nous avons indiqué ses temps de présence (des intervalles de la forme $[t_{apparition}, t_{disparition}]$, généralement un seul intervalle, mais parfois plusieurs si l'objet disparaît puis réapparaît plus tard). Nous avons ensuite attribué une catégorie à l'objet, ainsi qu'un identifiant partagé entre les caméras (pour des tâches de réidentification) : un même objet qui apparaît dans différentes caméras dispose du même identifiant. Enfin, nous avons tracé des boîtes englobantes régulièrement pendant toute la durée de présence de l'objet, généralement une par seconde, parfois davantage, de sorte à ce que la position à tout instant de la boîte englobante de l'objet puisse être interpolée linéairement en fonction des positions des boîtes englobantes précedente et suivante.

— Des annotations audio, pour la reconnaissance de sons, délimitées également par des intervalles de temps. Chaque annotation audio est également associée à un objet visuel par le biais de son identifiant : la voix d'une personne a ainsi le même identifiant que les boîtes englobantes correspondant à cette même personne, de même que le bruit de moteur d'une voiture est associé aux boîtes englobantes correspondant à cette voiture. Ce processus d'association a fait l'objet de nombreuses discussions autour de la notion d'objet audiovisuel et des liens entre annotations audio et annotations vidéo. Ces discussions ont donné lieu à une publication [Guyot 19] à laquelle nous sommes associés. Nous reviendrons dessus plus en détails dans le bilan de nos contributions au chapitre 6.

— Des annotations d'actions, pour faire de la détection d'évènements. Ces annotations recensent les temps de certains évènements (par exemple l'ouverture ou la fermeture d'une porte, le démarrage d'une voiture, ou une interaction entre deux personnes), et les identifiants des objets impliqués dans cette action.

Nous avons effectué la tâche d'annotation vidéo sur les 25 vidéos du premier scénario. Nous avons ensuite embauché une stagiaire pour annoter les 25 vidéos du second scénario sur le même modèle. Il existe des outils publics d'annotation de vidéos par tracé de boîtes englobantes (VATIC[3] en est un exemple). Cependant, aucun des outils disponibles ne dispose d'une fonctionnalité permettant d'attribuer un identifiant aux objets, pour pouvoir les réidentifier dans d'autres vidéos. De plus, aucun outil ne propose de conjointement

3. http://www.cs.columbia.edu/~vondrick/vatic/

annoter l'audio et la vidéo.

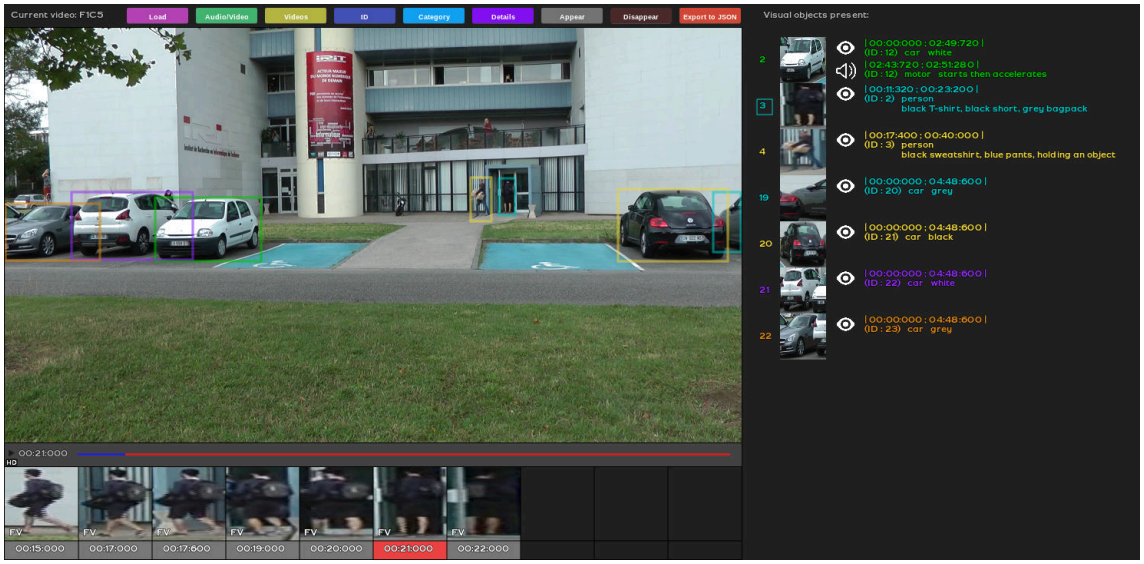


FIGURE 3.12 – Interface d'annotation du jeu de données ToCaDa — Interface développée pour l'annotation vidéo et audio multicaméra. Les boutons colorés en haut permettent respectivement de charger un fichier d'annotation déjà existant, d'alterner entre l'annotation audio et l'annotation vidéo, de choisir la vidéo à annoter, de modifier l'identifiant d'un objet, sa catégorie, d'indiquer des détails, de préciser à quel instant l'objet apparaît à l'écran, à quel instant il disparaît et enfin d'exporter le fichier d'annotation sous le format JSON. Dans la partie de droite, l'ensemble des objets apparaissant dans la vidéo sont affichés. Pour chaque objet, on retrouve l'imagette annotée la plus proche vis-à-vis de l'image courante et les informations relatives à l'annotation visuelle (symbolisée par un œil) et à l'annotation sonore lorsqu'il y en a une (symbolisée par un symbole de son). On retrouve notamment les intervalles respectifs de présence et d'audibilité. L'identifiant de l'objet en cours d'annotation est encadré (ici le 3). La partie centrale affiche la vidéo courante à l'instant courant. Il est possible d'y tracer une boîte englobante correspondant à la position courante de l'objet en cours d'annotation ou de modifier celles qui existent. Une barre de temps est présente en dessous et permet la lecture ou la mise en pause de la vidéo. Enfin, la barre en bas affiche les annotations déjà effectuées sur l'objet en cours d'annotation. Il est possible d'y cliquer pour se rendre directement à l'instant d'une annotation. Les lettres « OC » indiquent que l'objet est occulté et « FV » (*fully visible*) indiquent qu'il ne l'est pas.

Dans ce contexte, nous avons développé notre propre outil d'annotation, sous la forme d'une interface graphique interactive dans laquelle l'utilisateur peut tracer des boîtes englobantes, attribuer des identifiants et des catégories, ainsi qu'indiquer des intervalles de présence (pour la partie vidéo) ou d'audibilité (pour la partie audio). Enfin, pour chaque boîte englobante tracée, l'utilisateur peut indiquer si l'objet est occulté ou non. Une capture d'écran de l'interface est présentée sur la figure 3.12. Cette interface a été partagée avec les autres équipes, pour leur permettre d'enrichir nos annotations vidéo avec leurs annotations audio.

Le jeu de données ToCaDa a fait l'objet d'une publication à la conférence ACM Multimedia System (MMSys) à Amsterdam en juin 2018 [Malon 18] et a été mis en avant sur le site de ACM SIGMM [4].

3.3.3 Anonymisation des vidéos

Pour respecter les règles en termes de droit à l'image, nous avons dans un premier temps fait signer une demande d'autorisation de publication d'image à chaque participant. Soucieux des problématiques relatives à la protection de la vie privée, nous avons ensuite entrepris d'anonymiser les vidéos. Nous avons pour cela effacé les plaques d'immatriculation et appliqué un flou gaussien au niveau des visages.

En bilan des présentations des différents jeux de données existants et du nôtre, la figure 3.13 illustre les miniatures des vidéos que nous avons utilisées pour évaluer nos contributions ainsi que les liens de recouvrement qui existent entre elles. Maintenant que nous avons présenté l'ensemble des jeux de données répondant à nos besoins, nous allons présenter les expériences et les résultats de notre approche de recherche de liens par comparaisons d'histoires.

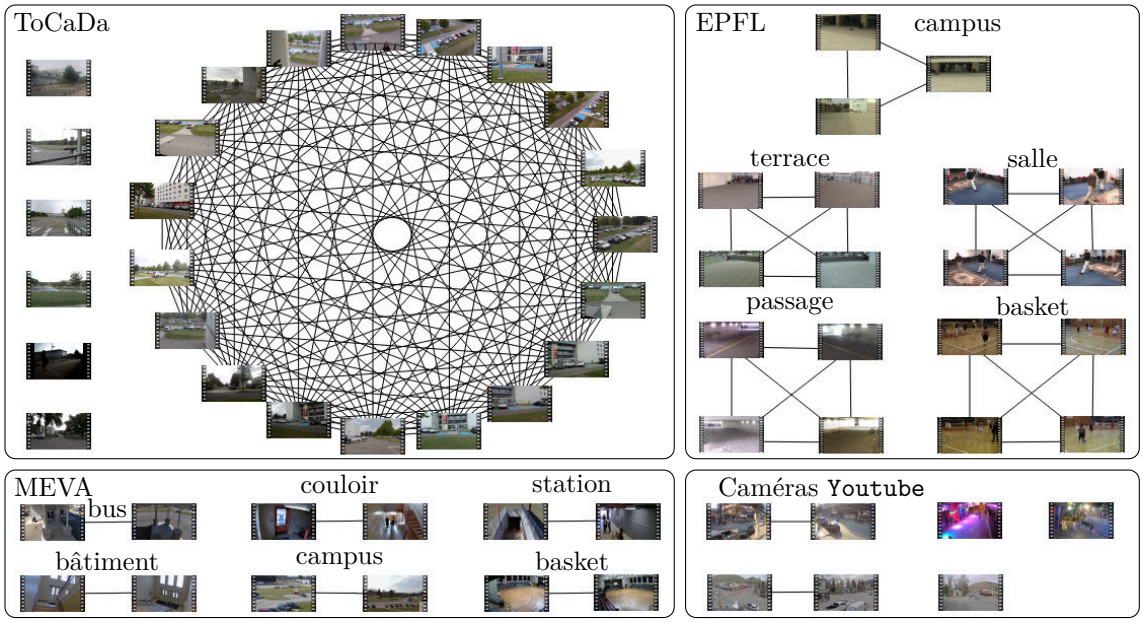


FIGURE 3.13 – Graphe de liens théorique des 63 vidéos utilisées dans nos expérimentations – Une arête entre deux vidéos indique qu'elles présentent du recouvrement dans leurs champs de vue. Les vidéos liées du jeu de données ToCaDa forment un graphe complet.

4. http://records.sigmm.org/2020/03/05/dataset-column-tocada-dataset-with-multi-viewpoint-synchronized-videos/

3.4 Évaluation de l'estimation des liens par comparaison d'histoires

Notre approche de recherche des liens par comparaison d'histoires exploite la cohérence globale entre détections d'objets pour former un graphe de liens entre vidéos. Bien que nous ne proposions pas de contribution en termes de détection d'objets, nous évaluerons dans un premier temps les résultats obtenus en termes de détections, car c'est à partir de ces détections qu'est calculé le graphe de liens. Trois méthodes de l'état de l'art, présentées au chapitre 2, sont évaluées, à savoir SSD [Liu 16], Mask-RCNN [He 17] et YOLOv3 [Redmon 18]. Pour chaque méthode, nous détectons les objets présents dans chaque vidéo tous les pas de temps $\tau = 500\,\text{ms}$. Les détections sont évaluées par la précision, le rappel et le score F_1 en considérant une boîte englobante comme bien détectée si la catégorie de l'objet est correctement déterminée et si l'indice de Jaccard de cette boîte englobante avec la boîte englobante de référence est supérieur à 0,5 (comme dans le *Pascal VOC challenge* [Everingham 10]). Par ailleurs, seules les détections avec une confiance supérieure à 0,5 sont gardées. Les résultats sont présentés dans le tableau 3.3. De manière générale, les trois méthodes font peu de fausses détections, ce qui se traduit par un rappel proche de 100%. En termes de précision, SSD présente des scores très inférieurs aux autres méthodes, car la plupart des vidéos des jeux de données utilisés contiennent des objets petits et nombreux, que l'algorithme ne parvient pas à détecter. YOLOv3 donne en moyenne les meilleurs scores. On remarque que Mask-RCNN obtient les meilleurs résultats sur les vidéos de haute résolution.

Évaluons à présent notre approche de formulation du graphe de liens de recouvrement. Trois niveaux de résolution spatiale et temporelle sont utilisés. À la résolution spatiale la plus grossière, on considère le cadre de la vidéo dans sa globalité. À la résolution la plus fine, on considère un quadrillage régulier de 9×9 cellules. La résolution temporelle varie de $\pm 3\tau$ à la résolution la plus grossière jusqu'à $\pm 1\tau$ à la résolution la plus fine. Pour chaque boîte englobante, trois descripteurs différents sont calculés sur l'imagette qu'elle contient. Le premier est le descripteur *ColorNames* [Yang 14]. Il découpe l'imagette en 6 tranches horizontales et associe à chaque tranche une distribution de 16 couleurs saillantes. Les descripteurs sont ensuite comparés en utilisant la mesure KISSME, *Keep It Simple and Straightforward MEtric* [Koestinger 12], décrite dans l'annexe B. Le deuxième descripteur employé est HOG, *Histogram of Oriented Gradients*, l'histogramme des gradients orientés [Dalal 05]. La mesure utilisée est une distance euclidienne. Enfin, le troisième type de descripteur est une représentation latente de l'imagette dans l'espace latent d'un réseau de neurones convolutif. Nous avons choisi le réseau ResNet18 [He 16], pré-entrainé sur la base d'images ImageNet [Deng 09]. Ces descripteurs sont comparés par la distance cosinus.

Nous calculons alors le graphe de liens en faisant varier les seuils d'apparence σ_{app} et

	Propriétés			SSD			Mask-RCNN			YOLOv3		
Jeu de données	#vid	Durée	Résolution	Prc	Rpl	F_1	Prc	Rpl	F_1	Prc	Rpl	F_1
Caméras `Youtube`	7	5:00	1920 × 1080	7,5	99,2	13,8	84,6	95,3	89,2	83,8	99,5	**90,8**
MEVA (global)	12		1920 × 1080	17,9	99,4	25,9	84,9	98,7	**90,9**	72,9	98,2	82,9
MEVA basket	2	5m00	1920 × 1080	1,9	100	3,6	85,8	98,6	**91,6**	58,1	98,3	73,0
MEVA couloir	2	5m00	1920 × 1080	61,9	100	76,4	92,9	100	**96,2**	91,0	100	95,2
MEVA bâtiment	2	5m00	1920 × 1080	19,6	100	32,7	73,3	100	**83,5**	61,7	95,5	73,9
MEVA bus	2	5m00	1920 × 1080	14,0	96,4	24,4	88,9	99,4	**93,7**	80,4	99,3	88,7
MEVA campus	2	5m00	1920 × 1080	4,2	100	7,7	83,0	100	**90,6**	64,7	96,1	76,8
MEVA station	2	5m00	1920 × 1080	5,8	100	10,4	85,8	94,1	**89,8**	81,4	100	89,6
EPFL (global)	19		360 × 288	54,4	99,2	67,1	74,7	95,4	83,4	92,6	99,5	**95,9**
EPFL salle	4	1m58	360 × 288	76,0	99,2	85,9	84,7	99,3	91,4	95,0	100	**97,4**
EPFL campus	3	3m55	360 × 288	74,4	98,9	84,9	83,8	97,8	90,2	95,3	100	**97,6**
EPFL basket	4	6m14	360 × 288	26,9	99,4	41,9	66,7	95,5	78,5	94,9	99,6	**97,2**
EPFL passage	4	1m40	360 × 288	28,2	100	43,5	61,8	85,1	70,2	90,2	99,1	**94,3**
EPFL terrasse	4	2m59	360 × 288	66,4	98,7	79,3	76,8	99,3	86,6	87,6	98,9	**92,9**
ToCaDa	25	4m48	960 × 540	13,5	99,8	22,4	74,8	96,5	83,1	80,5	99,4	**88,2**
Tous	63			25,7	99,5	35,3	77,7	96,4	85,2	83,0	99,2	**89,7**

Tableau 3.3 – Résultats des détections – Nous présentons également les propriétés des différents jeux de données : nombre (#vid), durée et résolution des vidéos. Pour l'évaluation de la détection, nous indiquons la précision moyenne (Prc), le rappel moyen (Rpl) et la moyenne des scores F_1. Les résultats sont donnés en pourcentage.

d'acceptation σ^1_{accept} entre 0 et 1 par pas de 0,05, et prenons $\sigma^s_{accept} = 1 - (1 - \sigma^1_{accept})^s$. σ^s_{rejet} est pris égal à $(1+\sigma^s_{accept})$ / 2. Plusieurs variantes de la méthode sont testées :

- en considérant toutes les détections simultanées, qu'importent la catégorie et l'apparence des objets ;
- en ne prenant en compte que les détections simultanées d'objets de la même catégorie ;
- en prenant en compte la catégorie et l'apparence des détections simultanées. Les trois types de descripteurs sont testés séparément.

Pour chaque paire de seuils (σ_{app}, σ^s_{accept}), nous comptabilisons les liens correctement trouvés (les vrais positifs), les mauvais liens trouvés (les faux positifs) et les liens manquants (les faux négatifs), puis nous calculons le score F_1 associé. Le tableau 3.4 indique les valeurs maximales de F_1 obtenues pour chaque jeu de données. Au niveau des détecteurs, Mask-RCNN et YOLOv3 présentent, comme pour les résultats de détection, de bons résultats, tandis que la méthode SSD donne des scores moins élevés. Nous expliquons ce résultat par la mauvaise précision de SSD pendant la phase de détection. De manière générale, la prise en compte de la catégorie améliore le score. L'effet est moindre sur les jeux de données EPFL et MEVA car ils contiennent essentiellement une seule catégorie d'objets (des personnes). L'utilisation de l'apparence est également bénéfique et, parmi les trois types de descripteurs testés, la représentation latente donne les meilleurs résultats.

	SSD					Mask-RCNN					YOLOv3				
Jeu de données	base	ctg	ctgcn	ctghg	ctglr	base	ctg	ctgcn	ctghg	ctglr	base	ctg	ctgcn	ctghg	ctglr
Caméras `Youtube`	19	22	27	25	25	80	**100**	**100**	**100**	**100**	80	**100**	**100**	**100**	**100**
MEVA (global)	28	29	32	30	35	59	60	67	62	**80**	59	59	65	68	76
EPFL (global)	56	56	64	59	67	61	61	71	65	79	63	63	74	66	**81**
ToCaDa	25	29	34	32	35	83	87	88	87	88	85	88	89	88	**90**
Tous	16	19	21	20	23	32	36	43	39	47	35	38	47	44	**51**

Tableau 3.4 – Scores F_1 les plus élevés des graphes de liens obtenus en fonction de trois méthodes de détection et les variantes de l'approche proposée. L'approche de base (base) prend en compte toutes les détections simultanées, peu importe la catégorie ou l'apparence. L'approche par catégorie uniquement (ctg) ne prend pas en compte l'apparence. Les approches ctgcn, ctghg et ctglr prennent en compte la catégorie et l'apparence en utilisant respectivement des descripteurs : Color Names, HoG et des descripteurs latents d'un réseau convolutif. Les résultats sont donnés en pourcentage.

3.5 Conclusion

À partir d'un ensemble de vidéos issues de caméras fixes couvrant la même période de temps, la méthode proposée parvient à estimer le graphe des liens de recouvrement entre vidéos. Il s'agit d'un graphe où chaque nœud correspond à un fichier vidéo et où les paires de vidéos dont les champs de vue se croisent sont liées par une arête. La méthode proposée s'appuie sur des détections concomitantes d'objets dans des paires de vidéos. Nous l'avons évaluée sur différents jeux de données en prenant en compte ou non la catégorie des objets détectés et en testant différents descripteurs d'apparence. Les graphes obtenus sur chaque jeu de données sont précis et le graphe obtenu sur l'union des jeux de données reste correct : les scores F_1 les plus élevés obtenus lors de l'évaluation individuelle par jeu de données vont de 0,79 à 1 pour *Mask-RCNN* et de 0,76 à 1 pour YOLOv3, tandis qu'ils sont de l'ordre de 0,5 lorsque tous les jeux de données sont utilisés à la fois. Les meilleurs résultats ont été obtenus en utilisant YOLOv3 pour la détection d'objet, et des descripteurs latents pour caractériser l'apparence. Ce travail a été accepté pour publication à la conférence ICPR 2020 [Malon 20b]. Notre approche fait intervenir plusieurs seuils. Pour des paires de cellules dont la distance est proche d'un seuil, le risque est de manquer un lien ou au contraire d'en détecter un à tort. Dans le contexte d'application d'une enquête, une erreur peut être très coûteuse car elle risque de donner de mauvaises indications voire d'orienter sur une fausse piste. Dans le chapitre suivant, nous cherchons donc à réduire ce risque en faisant intervenir l'utilisateur dans le processus d'apprentissage des liens par la machine.

Chapitre 4

Recherche de liens entre vidéos par apprentissage actif

Contenu

4.1 Introduction

Nous rappelons que l'objectif est de simplifier la tâche des opérateurs en automatisant certains traitements afin de réduire de manière significative leurs temps de recherche d'informations dans une collection de vidéos. La recherche complètement automatique introduite au chapitre précédent permet d'obtenir une estimation encourageante des liens entre vidéos. Toutefois, cette approche fait intervenir des seuils sur la distance entre les histoires de deux régions extraites des deux vidéos comparées. Cette distance doit être inférieure à un certain seuil pour que les régions soient considérées comme liées. Une légère modification des données d'entrée peut entraîner une variation de cette distance et faire manquer une correspondance. Ainsi, sur certains cas difficiles, notre approche peut déterminer des liens erronés ou ne pas déterminer certains liens corrects. Afin d'éviter au maximum ce type d'erreurs, on peut mettre en place un mécanisme de sollicitation de l'utilisateur pour aider la machine à prendre certaines décisions. Ainsi, au lieu de laisser la machine faire tous les traitements au risque qu'elle se trompe, l'homme et la machine peuvent coopérer et être plus efficaces qu'ils ne l'auraient été séparément. Cette famille de méthode d'apprentissage porte le nom d'**apprentissage actif** et peut être vue comme de l'apprentissage semi-supervisé dans le sens où l'algorithme ne dispose pas de toutes les étiquettes en entrée mais seulement d'un sous-ensemble (voire d'aucune). Dans ce chapitre, nous commençons par rappeler les différentes notions liées à l'apprentissage (apprentissage supervisé, apprentissage non supervisé et apprentissage actif) avant de présenter une approche utilisant l'apprentissage actif dans le contexte de la recherche de liens dans une collection de vidéos.

De nombreux indices peuvent aider à valider ou écarter l'existence d'un lien entre deux vidéos. Il peut s'agir de la similarité de l'arrière-plan des vidéos, comme présenté dans la section 2.4, de cooccurrences d'objets présents à l'image, voir section 2.5.2, ou même d'un son entendu simultanément dans les vidéos, comme vu dans le cadre du projet VICTORIA.Toutes ces caractéristiques, que l'on peut extraire par des algorithmes, permettent de décrire les vidéos. C'est à partir de ces caractéristiques, multiples et variées, que l'on souhaite déterminer l'existence ou non de liens entre vidéos. Pour ce faire, différentes méthodes de classification, basées sur de l'apprentissage, peuvent être employées. Nous en détaillons le formalisme et décrivons les approches les plus fréquemment utilisées dans les sections 4.2, 4.3 et 4.4. Nous proposons ensuite une approche d'apprentissage actif impliquant l'utilisateur pour aider la machine dans son apprentissage sur les données difficiles à classifier. Nous présentons ensuite nos expérimentations, ainsi qu'un prototype d'interface mettant en œuvre notre approche. Nous omettons volontairement de présenter les approches faisant appel à l'apprentissage profond. En effet, elles demandent généralement un nombre important de données correspondant au problème traité (dans notre cas des vidéos dont les champs de vue se recouvrent), ce dont nous ne disposons pas.

Supposons que l'on dispose de m individus, chacun étant décrit par n variables, stockés dans une matrice $\mathbf{X} \in \mathbb{R}^{m \times n}$. Pour chaque individu en entrée $\mathbf{x}_i$, on souhaite déterminer en sortie y_i tel que :

— $y_i \in \mathbb{R}$ lorsque la sortie que l'on cherche à estimer est une valeur dans un ensemble réel continu (par exemple si l'on cherche à estimer la probabilité d'existence d'un lien, on aura $y_i \in [0,\ 1]$). On parle alors de **régression** ;

— $y_i \in \{c_1, \ldots, c_N\}$ où les c_i sont des étiquettes de classe si l'ensemble des valeurs possibles est fini. On dit alors qu'il s'agit d'un problème de **classification**. C'est ce type de problème que nous traitons. Ainsi, pour la suite de ce chapitre, nous ne présentons que les approches supervisées dans le contexte de la classification.

On distingue différentes familles de méthodes d'apprentissage en fonction du degré de supervision. La figure 4.1 illustre les familles les plus courantes.

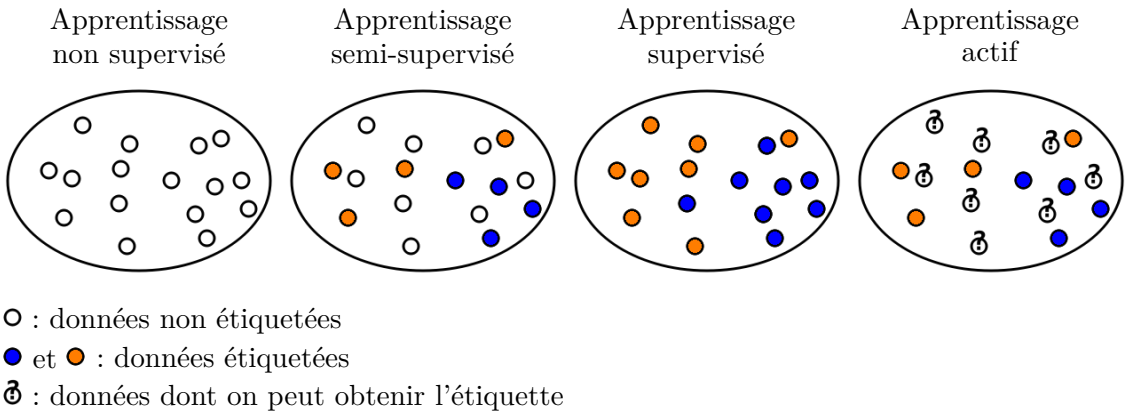


FIGURE 4.1 – Différents degrés de supervision en apprentissage – Exemple à deux classes représentées en bleu et orange. L'apprentissage est qualifié de non supervisé si l'on ne dispose d'aucune étiquette, de semi-supervisé si l'on dispose de certaines étiquettes et de supervisé si l'on dispose de toutes les étiquettes. Enfin, si l'on est en mesure de demander à un expert de donner une étiquette manquante, on parle d'apprentissage actif.

4.2 Apprentissage supervisé

Une tâche d'apprentissage est dite supervisée si l'on dispose des sorties y_i pour toute entrée $\mathbf{x}_i$. Le but est alors, à partir de cet ensemble d'apprentissage, de construire un modèle m qui permette, pour une nouvelle entrée $\mathbf{x}$, de déterminer sa sortie y, c'est-à-dire un mécanisme de prédiction. Il existe de nombreuses méthodes d'apprentissage supervisé et nous présentons brièvement les approches les plus utilisées : l'estimation bayésienne dans la section 4.2.1, l'approche des k plus proches voisins dans la section 4.2.2, la méthode par séparateurs à vaste marge (SVM) dans la section 4.2.3 et enfin celle des arbres de décision dans la section 4.2.4.

4.2.1 Estimation bayésienne

Pour chaque classe c_j, l'ensemble des points $\mathbf{x}_i$ appartenant à cette classe (c'est-à-dire tels que $y_i = c_j$) forme un nuage de points qui peut être modélisé par une loi normale multidimensionnelle. La densité de probabilité d'une loi normale de moyenne μ et de matrice de covariance Σ, notée $\mathcal{N}(\mu, \Sigma)$, s'écrit, en dimension n :

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right]. \tag{4.1}$$

Dans le cadre bayésien, nous supposons que la vraisemblance de chaque classe c_j, caractérisée par la moyenne μ_j et la matrice de covariance Σ_j, suit une loi normale qui, de façon similaire à celle de la densité de probabilité définie en (4.1), s'écrit :

$$p(\mathbf{x}|c_j) = \frac{1}{(2\pi)^{n/2}|\Sigma_j|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_j)^{\mathrm{T}}\Sigma_j^{-1}(\mathbf{x}-\mu_j)\right]. \tag{4.2}$$

Lorsque l'on souhaite classer une nouvelle donnée $\mathbf{x}$, on peut alors procéder par **maximum de vraisemblance** : pour cette entrée $\mathbf{x}$, on calcule la vraisemblance $p(\mathbf{x}|c_j)$ pour chacune des classes c_j, et on associe $\mathbf{x}$ à la classe de plus forte vraisemblance. Par ailleurs, la règle de Bayes donne l'expression suivante de la probabilité *a posteriori* $p(c_j|\mathbf{x})$, c'est-à-dire de la probabilité pour que la classe c_j contienne $\mathbf{x}$:

$$p(c_j|\mathbf{x}) = \frac{p(\mathbf{x}|c_j)p(c_j)}{p(\mathbf{x})}. \tag{4.3}$$

Lorsque l'on assigne $\mathbf{x}$ à la classe qui maximise $p(c_j|\mathbf{x})$, on parle de classification par **maximum *a posteriori***. Sachant que le dénominateur $p(\mathbf{x})$ de (4.3) est indépendant de c_j, il n'est pas nécessaire de le connaître pour trouver le maximum des $p(c_j|\mathbf{x})$. En revanche, il est nécessaire de connaître la probabilité *a priori* $p(c_j)$ de chaque classe c_j, faute de quoi on fait généralement l'hypothèse que les classes sont équiprobables. La classification par maximum *a posteriori* revient alors à une classification par maximum de vraisemblance.

4.2.2 Méthode des k plus proches voisins

La méthode des k plus proches voisins [Cover 67] consiste à assigner à une donnée $\mathbf{x}$ l'étiquette la plus représentée parmi les k données d'apprentissage $\mathbf{x}_i$ les plus proches de $\mathbf{x}$. L'algorithme 2 présente la méthode en pseudo-code. Si $k = 1$, la méthode revient à assigner à $\mathbf{x}$ l'étiquette de la donnée d'apprentissage $\mathbf{x}_i$ la plus proche. Cette méthode est simple à mettre en œuvre et ne nécessite qu'un seul paramètre : le nombre de voisins à considérer k. Cependant, ce paramètre peut être difficile à choisir. Par ailleurs, la méthode est sensible aux données aberrantes dont l'étiquette est fausse.

Algorithme 2 : Algorithme des k plus proches voisins

Données :
$\mathbf{X} = (\mathbf{x}_i, y_i)$ données étiquetées
$k \in \mathbb{N}$ nombre de voisins
$\mathbf{x}$ donnée à étiqueter
Résultat : $\hat{y}$ classe à laquelle assigner $\mathbf{x}$

1 Calculer L la liste des distances entre $\mathbf{x}$ et $\mathbf{x}_i$
2 Trier L par ordre croissant
3 Appliquer les mêmes changements d'indices aux éléments de $\mathbf{X}$
4 Récupérer les étiquettes $\{e_1, ..., e_k\}$ des k premiers éléments de $\mathbf{X}$
5 Retourner la classe la plus représentée dans $\{e_1, ..., e_k\}$

4.2.3 Séparateurs à vaste marge (SVM)

Pour traiter les cas où les données ne sont pas linéairement séparables, SVM propose de transformer l'espace de représentation des données d'entrée $\mathbf{x}_i$ en un espace de plus grande dimension dans lequel il est possible de trouver une séparation linéaire. Cette transformation se fait par le biais d'une **fonction de noyau**. Il peut s'agir par exemple d'une fonction polynomiale. Une fois les données linéairement séparables, SVM sélectionne l'hyperplan séparateur de marge maximale, c'est-à-dire celui qui maximise la distance des $\mathbf{x}_i$ à cet hyperplan. La figure 4.2 illustre l'idée générale de cette approche. En pratique, trouver une fonction de noyau qui permette de séparer linéairement l'ensemble des données n'est pas trivial et il n'y a pas de solution absolue. En général, certaines données se retrouvent mal classées. Pour prendre en compte ces erreurs, les auteurs de [Cortes 95] proposent d'utiliser des marges souples, robustes aux données mal classées, en cherchant un hyperplan séparateur qui minimise le nombre d'erreurs.

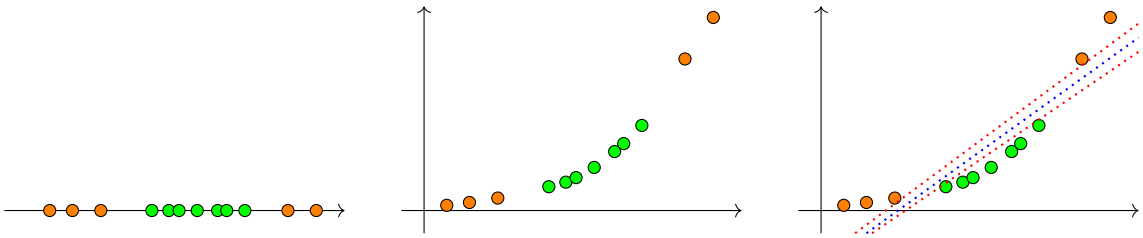


FIGURE 4.2 – Illustration de l'algorithme de classification par SVM – Nous montrons un ensemble de données formé de deux classes non linéairement séparables dans un espace de dimension 1 (a). En transformant l'espace de représentation des données en un espace de dimension supérieure (b), les données deviennent linéairement séparables (c). Parmi les différents hyperplans séparateurs possibles dont quelques uns sont représentés en pointillés, SVM choisit celui de marge maximale, en bleu. Dans cet exemple, la fonction de noyau utilisée est polynomiale de degré 2.

4.2.4 Arbres de décision

L'apprentissage par arbre de décision s'appuie sur la structure hiérarchique des arbres pour choisir la classe à assigner à une donnée en fonction de ses différentes variables. Partant du tronc de l'arbre, la donnée circule de nœud en nœud à travers les branches de l'arbre jusqu'à arriver à l'une des feuilles qui indique la classe à assigner. À chaque nœud, le choix de la branche à emprunter dépend des valeurs d'une ou plusieurs variables de la donnée. Généralement, les algorithmes de construction d'arbres de décision divisent l'arbre du sommet vers les feuilles en sélectionnant à chaque étape la variable d'entrée qui réalise le meilleur partage de l'ensemble des données selon un critère donné. Intuitivement, il s'agit du partage qui, le mieux possible, sépare les données de classes différentes dans des nœuds-fils différents et qui regroupe les données d'une même classe dans un même nœud-fils.

Cette section nous a permis de faire un inventaire rapide des approches d'apprentissage supervisé les plus utilisées pour lesquelles nous donnons peu de détails puisqu'il ne s'agit pas du type d'approches que nous pouvons utiliser. À présent, nous introduisons deux des techniques les plus connues en apprentissage non supervisé.

4.3 Apprentissage non supervisé

Lorsque l'on cherche à apprendre et que les données ne sont pas étiquetées, c'est-à-dire lorsqu'on ne dispose pas des y_i, on parle d'apprentissage non supervisé. L'idée est alors de former des classes d'individus présentant des caractéristiques d'entrée $\mathbf{x}_i$ proches. Le nombre de classes peut être connu à l'avance, comme dans l'approche par k-moyennes que nous détaillons dans la section 4.3.1 (par exemple deux classes { liées, non liées }) ou non, comme dans la méthode *mean-shift*, détaillée dans la section 4.3.2. Ces deux méthodes sont des méthodes de **partitionnement**.

4.3.1 Méthode des k-moyennes

Proposée dans [MacQueen 67], cette méthode suppose que l'on connaît à l'avance le nombre de classes k. L'algorithme 3 en présente les différentes étapes.

La sélection des germes peut être aléatoire ou tenir compte de la répartition des données. Par exemple, la version k-moyennes++ [Arthur 06] propose une initialisation améliorant la probabilité d'obtenir la solution optimale en éloignant au maximum les k points des moyennes initiales. Dans cette variante, le germe de la première classe est choisi aléatoirement parmi les données, puis chaque autre germe est choisi parmi les autres données, avec une probabilité proportionnelle au carré de la distance entre la donnée candidate pour devenir un germe et le germe le plus proche.

Algorithme 3 : Algorithme des k-moyennes

Données :
$\mathbf{X}$ données non étiquetées contenant les points $\mathbf{x}_i \in \mathbb{R}^n$
$k \in \mathbb{N}$ nombre de classes
Résultat : Y vecteur des étiquettes de $\mathbf{X}$

Initialiser k centres de classes à k données distinctes de $\mathbf{X}$ (les germes)
Répéter
    Assigner à chaque donnée la classe du centre le plus proche suivant un critère
    Le centre de chaque classe devient la moyenne des points $\mathbf{x}_i$ de la classe
Jusqu'à ce que *les classes des données ne changent plus*
Retourner les classes assignées à chaque donnée à la dernière itération.

Plutôt que d'illustrer différentes itérations successives sur un exemple, nous proposons une interface interactivei que nous avons codé, accessible en ligne [1], qui permet de choisir le nombre de données et le nombre de classes k, puis met à jour dynamiquement la position des centres des classes. Une capture d'écran de cette interface est présentée sur la figure 4.3. Les données y sont caractérisées par leur position mais dans le cas des images et des vidéos, on utilise généralement des caractéristiques photométriques.

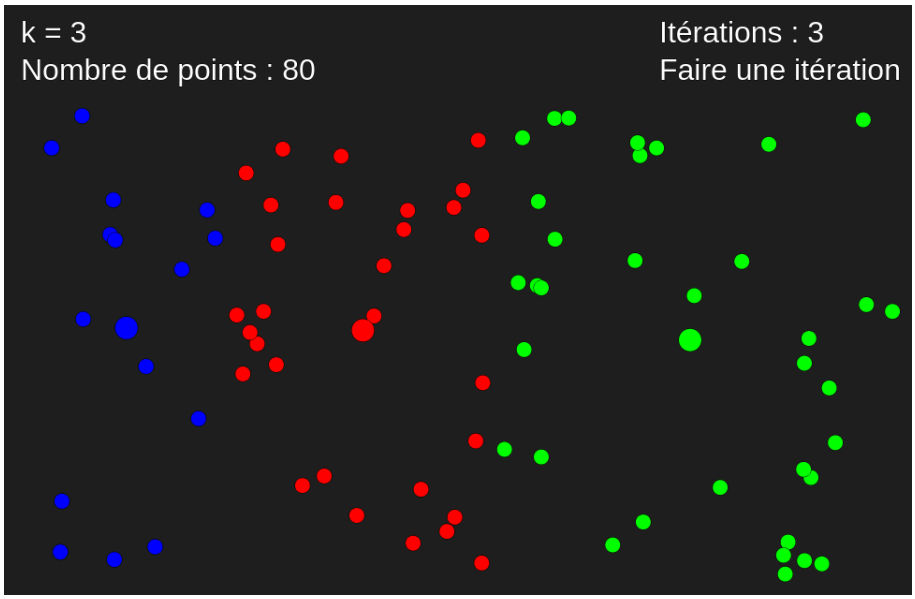


FIGURE 4.3 – Interface interactive de démonstration des k-moyennes — Pour illustrer cette approche, nous proposons une interface interactive dans laquelle l'utilisateur peut choisir le nombre de points et la valeur du nombre de classes k. Il peut ensuite lancer l'algorithme et effectuer une itération à chaque clic sur « Faire une itération ». Les mises à jour se font de façon dynamique. Chaque couleur correspond à une classe. Les gros points correspondent aux centres des classes. Dans cet exemple, le critère utilisé est un critère de proximité en distance euclidienne.

1. https://keurstudio.fr/Kmeans/

4.3.2 Méthode *mean-shift*

Contrairement à k-moyennes, l'approche *mean-shift* [Comaniciu 02] ne nécessite pas de connaître le nombre de classes. Elle permet de le déterminer à partir d'un paramètre de bande-passante r indiquant la distance maximale entre les données à regrouper. Il s'agit d'une méthode de recherche des modes d'une densité de probabilité dont les données sont des échantillons. L'algorithme 4 en décrit le fonctionnement en détails et consiste en les étapes suivantes :

1. chaque donnée $\mathbf{x}_i$ est dupliquée en $\mathbf{x}'_i$;
2. chaque donnée $\mathbf{x}'_i$ devient itérativement la moyenne des $\mathbf{x}'_j$ tels que $\mathrm{K}(\mathbf{x}'_i - \mathbf{x}'_j) < r$, où K est une fonction de noyau ;
3. l'algorithme prend fin lorsque toutes les données dupliquées ont convergé, et les données d'origine sont alors classées en fonction du mode vers lequel a convergé sa donnée dupliquée ; le nombre de modes correspond alors au nombre de classes et la classification est opérée de la façon suivante : deux données d'entrée $\mathbf{x}_i$ et $\mathbf{x}_j$ sont associées à la même classe si $\mathbf{x}'_i$ et $\mathbf{x}'_j$ convergent vers le même mode.

Algorithme 4 : Algorithme *mean-shift*

Données :
$\mathbf{X}$ données non étiquetées
$r \in \mathbb{R}$ bande passante
Résultat : Y vecteur des étiquettes de $\mathbf{X}$

```
Dupliquer X en X'
Répéter
    ∀i, x'_i devient la moyenne des x'_j vérifiant K(x'_i − x'_j) < r
Jusqu'à ce que ∀i, x'_i ait été peu modifié au cours de l'itération
Assigner la même classe aux x_i dont les x'_i sont proches
Retourner les classes assignées
```

Nous proposons une interface interactive similaire à celle illustrant la méthode des k-moyennes pour *mean-shift* accessible en ligne [2]. Cette fois, l'utilisateur peut régler le paramètre de bande-passante r en plus du nombre de points. Les itérations successives mettent en évidence la concentration des points autour d'un nombre limité de modes, qui représentent les différentes classes. Une capture d'écran de cette interface est présentée sur la figure 4.4. Encore une fois, c'est la position qui est utilisée, mais, dans le cas des images et des vidéos, on utilise également l'information photométrique.

Cette section nous a permis de présenter les outils principaux de l'état de l'art que nous utiliserons par la suite en tant qu'approches de base à comparer avec l'approche que nous proposons.

2. https://keurstudio.fr/Meanshift/

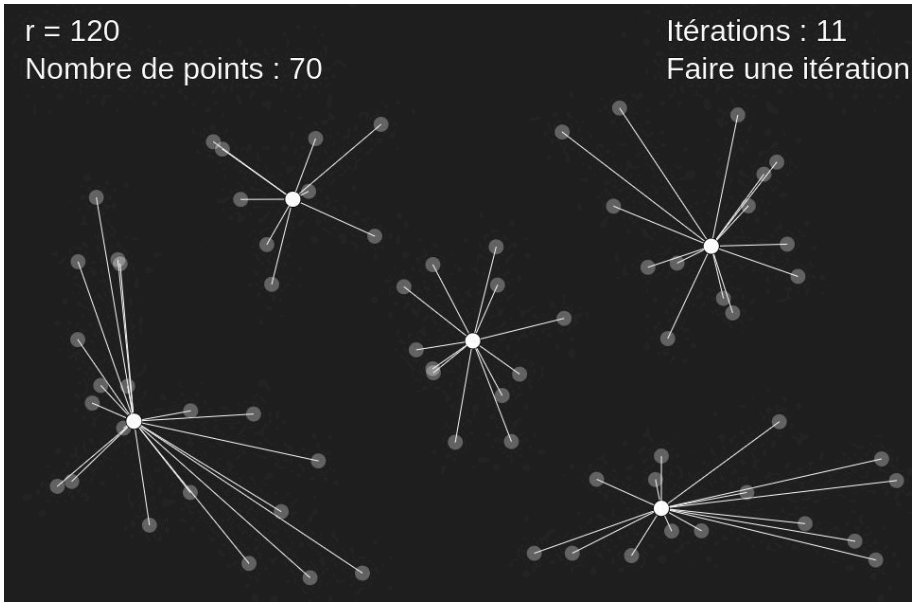


FIGURE 4.4 – Interface interactive de démonstration de *mean-shift* — Au travers de l'interface que nous proposons, l'utilisateur peut choisir le nombre de points et la valeur du paramètre de bande passant r. Il peut ensuite lancer l'algorithme *mean-shift* et effectuer une itération à chaque clic sur « Faire une itération ». Les mises à jour se font de façon dynamique. La position d'origine des points est représentée en transparence et un segment lie chaque point d'origine $\mathbf{x}_i$ à la position de son point dupliqué $\mathbf{x}'_i$.

4.4 Apprentissage actif

4.4.1 Lien avec l'apprentissage semi-supervisé

À mi-chemin entre apprentissage non supervisé et apprentissage supervisé se trouve l'apprentissage semi-supervisé. Dans ce cas de figure, on ne dispose que d'une partie des étiquettes des données d'entrée $\mathbf{x}_i$. Grâce à ces données, il est possible d'exploiter les données non étiquetées et de prédire leurs étiquettes. Le modèle, initialisé avec les données étiquetées, peut être mis à jour avec les données non étiquetées prédites avec un haut degré de confiance [Zhu 05]. Pour les données dont le modèle a du mal à prédire l'étiquette, une sous-famille de méthodes d'apprentissage semi-supervisé consiste à demander directement à un expert d'étiqueter. On parle alors d'apprentissage actif.

4.4.2 Principes de l'apprentissage actif

L'apprentissage actif consiste à considérer que l'on dispose :

— d'un ensemble de données étiquetées $\mathcal{L}$ à partir duquel on peut calculer un modèle h ;

— d'un ensemble de données non étiquetées $\mathcal{U}$ sur lequel on souhaite effectuer une tâche

de régression ou de classification ;

— d'un expert, **l'oracle**, capable d'étiqueter n'importe laquelle des données de $\mathcal{U}$.

Le modèle peut être mis à jour lorsqu'une donnée de $\mathcal{U}$ est étiquetée par l'oracle ou prédite par le modèle. La probabilité d'associer l'étiquette y à une donnée $\mathbf{x} \in \mathcal{U}$ vis-à-vis du modèle h sera notée $p_h(y|\mathbf{x})$. On notera également $\hat{y}(\mathbf{x})$ la sortie vérifiant :

$$\hat{y} = \underset{y}{\operatorname{argmax}}\ p_h(y|\mathbf{x}). \tag{4.4}$$

Pour toute donnée de l'ensemble $\mathcal{U}$, l'oracle peut être consulté. Une approche naïve consisterait alors à demander à l'oracle d'étiqueter toutes les données de $\mathcal{U}$. Appliqué à notre cas, cela reviendrait à demander aux opérateurs de déterminer les liens entre chaque paire de vidéos, ce que l'on cherche justement à éviter. De manière générale, pour des raisons de coûts, le but est au contraire de profiter de l'expertise de l'oracle le moins possible et d'en profiter le plus efficacement possible. Intuitivement, lorsque l'on cherche à apprendre quelque chose, on a besoin de l'aide d'un expert au début pour avoir des premiers exemples, puis, à mesure que l'on progresse, on le sollicite de moins en moins et sur des cas différents de ceux que l'on a déjà rencontrés.

Afin de choisir pour quelles données il est pertinent de solliciter l'oracle, plusieurs critères ont été proposés [Settles 09] et nous en exposons quelques-uns ci-après.

— On peut utiliser un critère d'**incertitude** comme introduit dans [Lewis 94] : dans ce cas, l'oracle est sollicité pour les données pour lesquelles le modèle est peu confiant, par exemple si la valeur de $p_m(\hat{y}|\mathbf{x})$ est inférieure à un certain seuil ou à l'inverse, lorsque plusieurs étiquettes ont une probabilité élevée d'être associées à $\mathbf{x}$. Par exemple, les auteurs de [Scheffer 01] proposent de prendre en compte l'écart entre les probabilités d'étiquetage des deux classes les plus probables $\hat{y}$ et $\hat{y}'$:

$$\sigma = p_h(\hat{y}|\mathbf{x}) - p_h(\hat{y}'|\mathbf{x}), \tag{4.5}$$

et de solliciter l'oracle lorsque cet écart est proche de 0.

— On peut utiliser un critère de **vote par comité** comme introduit dans [Seung 92] : dans ce cas, plusieurs modèles sont utilisés simultanément pour prédire l'étiquette des données — généralement avec des paramètres différents — et un mécanisme de vote est employé. L'oracle est alors sollicité pour étiqueter les données sur lesquelles les votes des différents modèles s'accordent le moins.

— On peut utiliser un critère de **modification maximale du modèle** comme introduit dans [Settles 08] : dans ce cas, l'idée est de déterminer les données dont la connaissance de l'étiquette apporterait les changements les plus importants aux paramètres du modèle et de solliciter l'oracle pour ces données. L'idée sous-jacente est que les données qui modifient le plus un modèle correspondent généralement aux

données les plus différentes de celles que le modèle a déjà rencontré et sont donc celles qui apportent le plus d'information.

— Enfin, on peut utiliser un critère de **modification maximale des sorties du modèle** : dans ce cas, on cherche à déterminer les données dont l'étiquetage apporterait les changements les plus importants aux futures prédictions.

En complément des critères considérés, différentes stratégies de sollicitation de l'oracle peuvent être adoptées [Settles 09].

— Dans la **stratégie au fil de l'eau**, on considère les données d'entrée une par une. Pour chaque donnée, la décision de solliciter ou non l'oracle est prise en fonction du modèle et du critère considéré. Ce type de stratégie, qui ne considère qu'une donnée à la fois, est peu coûteux, mais ne prend pas en compte la distribution des données d'entrée et risque de ne pas explorer équitablement l'espace des données.

— La **stratégie par paquets** considère toutes les données d'entrée et associe à chacune un score en fonction du modèle et du critère considéré. L'oracle est alors sollicité pour annoter les N données dont les scores sont les plus élevés. Cette stratégie est plus coûteuse, car elle sollicite l'oracle à N reprises pour chaque itération, mais a tendance à donner les meilleurs résultats.

— Enfin, la **stratégie de synthèse** cherche à générer des données d'entrée $\mathbf{x}_G$ telles que l'étiquetage par l'oracle y_G maximise le critère considéré. Le principal défaut de cette approche est qu'elle sollicite l'oracle sur des données synthétisées qui risquent de ne pas ressembler à des données réelles. Cela peut mettre en difficulté l'expertise de l'oracle. Dans les premiers travaux qui utilisent ce type de stratégie [Baum 91], les auteurs ont notamment essayé d'apprendre à un réseau de neurones à reconnaître de l'écriture manuscrite et ils indiquent que les lettres synthétisées étaient incompréhensibles pour des humains. Cependant, les récentes avancées des réseaux de neurones antagonistes génératifs (GAN) pour la génération de données de synthèse sont encourageantes pour la génération d'images réalistes et des travaux plus récents laissent entrevoir la possibilité de les coupler à l'apprentissage actif [Sinha 19].

Les deux premières stratégies sont qualifiées de **sélectives** car elles utilisent des données existantes, tandis que la troisième est qualifiée de **constructive** car elle génère une nouvelle donnée.

4.4.3 Travaux connexes

Avant de présenter nos travaux en lien avec l'apprentissage actif, nous présentons cinq domaines où l'apprentissage actif est utilisé.

En **imagerie médicale**, l'apprentissage actif est notamment utilisé pour la classification des différents types de tissus [Mahapatra 13]. Dans ce domaine, l'annotation des images

par un opérateur humain demande généralement un temps considérable du fait de la multitude de classes à considérer et de la résolution des images.

Pour la **reconnaissance d'objet**, où les jeux de données sont généralement faciles à produire mais coûteux à annoter, l'apprentissage actif peut permettre de produire rapidement des annotations en prédisant les classes des objets pour lesquelles le modèle est confiant et en sollicitant l'oracle sur les autres images. Les auteurs de [Joshi 09] ont notamment utilisé l'apprentissage actif sur 101 classes d'objets issues du jeu de données Caltech-101 [Fei-Fei 04] en comparant les performances de reconnaissance en fonction du nombre de sollicitations de l'oracle.

L'apprentissage actif est souvent inclus dans les **systèmes de recommandation**. Certaines plateformes multimédia proposent à leurs utilisateurs des contenus ciblés en fonction de leurs habitudes de consommation de contenu, par exemple sur la plateforme de partage de vidéos `Youtube`. Ces habitudes, apprises grâce aux vidéos visionnées par l'utilisateur, permettent de créer un modèle cherchant à prédire quelles vidéos ont la plus forte probabilité d'intéresser l'utilisateur. De plus, la plateforme peut solliciter directement l'utilisateur pour obtenir son avis sur une vidéo qu'il a visionnée afin d'affiner son modèle : il s'agit alors d'un retour sur la pertinence d'un contenu proposé (*relevance feedback*).

Plusieurs travaux intègrent l'apprentissage actif pour des tâches de classification de contenus multimédia. Les auteurs de [Sabata 18] cherchent à classer différentes séquences issues de séries télévisées en fonction de l'appartement dans lequel elles ont été tournées. Ils exploitent pour cela la colorimétrie des images en les décrivant par des histogrammes de couleurs. Ils utilisent l'apprentissage actif en sollicitant l'oracle pour les cas où le modèle ne parvient pas à prédire une classe avec une probabilité assez élevée. Sinon, ils assignent la classe la plus probable à la donnée et l'ajoutent à l'ensemble d'apprentissage si la probabilité est élevée. Le même genre d'approche est utilisé par [Karlos 19] sur différents types de problèmes et de modalités : l'identification du genre, la reconnaissance d'émotions et la détection de grossièreté.

Dans un contexte de surveillance, les auteurs de [Chen 11a] cherchent à estimer la topologie d'un réseau de caméras, comme illustré sur la figure 4.5, par réidentification de piétons et sollicitations de l'oracle. Ils considèrent deux types de modèles : un modèle local qui estime l'intérêt qu'apporterait l'annotation de chaque image individuellement et un modèle relationnel qui prend en compte un voisinage temporel dans son estimation de l'intérêt des images. Si l'identité de l'un des piétons est difficile à prédire sur une image donnée, le modèle local donnera un fort intérêt à ce que cette image soit annotée par l'oracle. En revanche, si l'identité de ce piéton est facile à prédire sur les images qui précèdent ou suivent, le modèle relationnel estimera qu'il y a peu d'intérêt à solliciter l'oracle. Ces deux modèles permettent d'identifier les images de la vidéo qui apporteraient le plus d'information si elles étaient annotées.

Dans [Khoshrou 14], les auteurs cherchent à proposer un résumé des activités dans

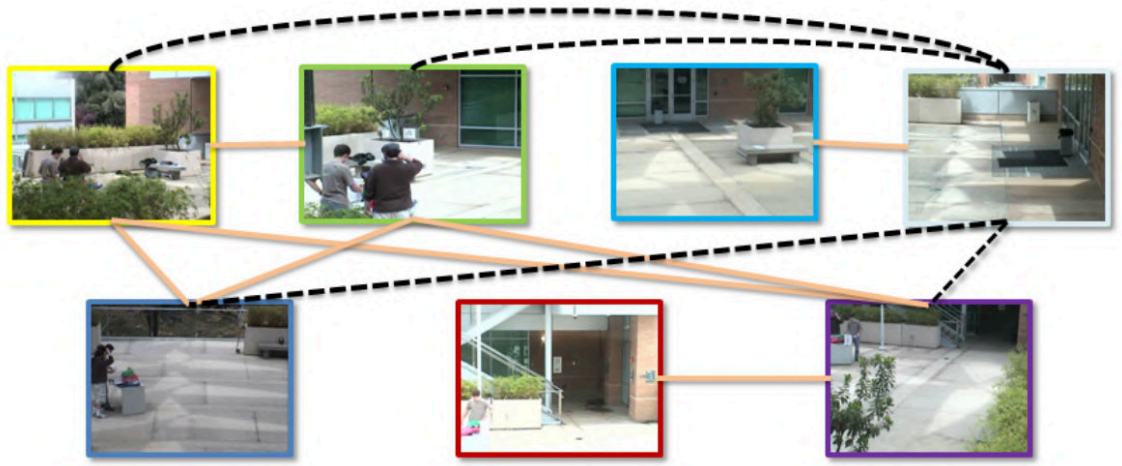


FIGURE 4.5 – Estimation de la topologie d'un réseau de caméras par apprentissage actif — À partir de réidentifications dans un réseau de 7 caméras et en sollicitant l'oracle pour les cas les plus difficiles, [Chen 11a] parviennent à estimer la topologie du réseau de caméras. Les liens en beige indiquent des vues où les piétons détectés sont corrélés tandis que les pointillés indiquent une corrélation négative.

un réseau de caméras. Ils constatent que sur les nombreuses heures enregistrées par les caméras de surveillance, seule une faible proportion présente de l'intérêt. Ils proposent alors de résumer les heures d'enregistrement par un diagramme temporel indiquant, pour chaque caméra, les temps de présence des différents objets ayant traversé le champ de vue de la caméra, comme représenté sur la figure 4.6. Le nombre d'objets n'est pas connu *a priori*. Ainsi, l'oracle peut être sollicité soit pour classer une donnée dans l'une des classes existantes, par exemple pour réidentifier un même objet apparaissant dans deux caméras différentes, soit pour confirmer ou non qu'un objet appartient à une nouvelle classe. Nous nous inspirerons de ce concept permettant à l'oracle de créer dynamiquement des classes dans notre contribution sur la recherche de liens entre vidéos à partir d'apprentissage actif.

4.5 Application à la recherche de liens entre vidéos

Dans le contexte des travaux de cette thèse, nous disposons d'un ensemble de m individus (les vidéos) que l'on peut caractériser par des descripteurs qui exploitent le contenu de la vidéo en termes d'objets qui y apparaissent et en termes d'arrière-plan. Ces descripteurs sont stockés sous forme de vecteurs de $\mathbb{R}^n$. Pour chaque paire de vidéos, on cherche à déterminer l'existence ou non d'un lien. Nous proposons deux façons différentes de modéliser le problème :

— un problème de classification à deux classes { liées, non liées }, où chaque vidéo est

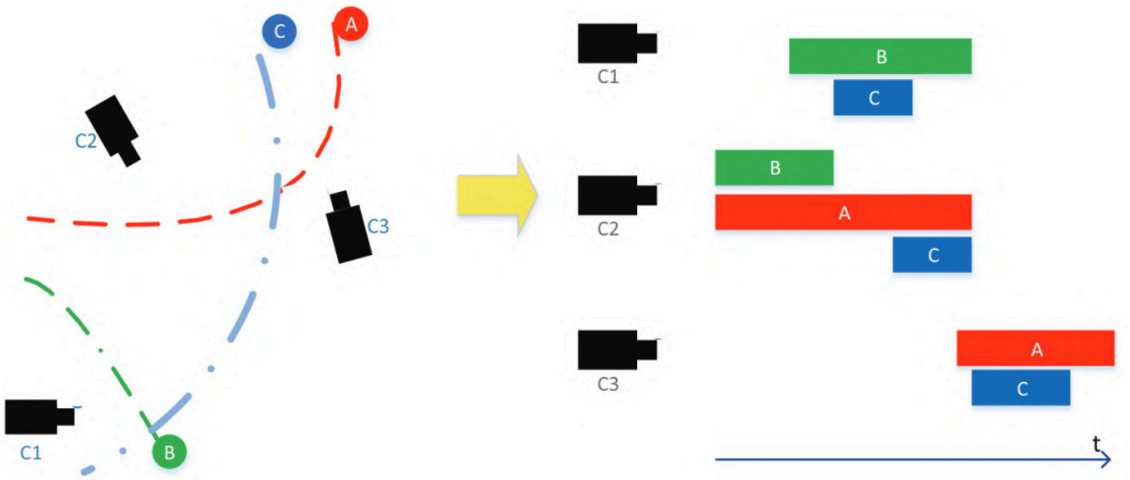


FIGURE 4.6 – Résumé des activités dans un réseau de caméras par apprentissage actif — À gauche, trois objets A (rouge), B (vert) et C (bleu) passent dans les champs de vue de trois caméras C_1, C_2 et C_3 sur une période de temps donnée. L'objectif des travaux de [Khoshrou 14] est d'offrir un résumé des apparitions dans le temps des différents objets dans les champs de vue des caméras sous forme d'un diagramme temporel, comme illustré à droite.

classée relativement à une vidéo d'intérêt donnée ;

— un problème de classification à N classes $\{c_1, \ldots, c_N\}$ où chaque classe représente un ensemble de vidéos formant un graphe connexe de vidéos dont les champs de vue des caméras se recoupent.

En termes de stratégie de sollicitation de l'oracle, la stratégie de synthèse n'est pas utilisable dans notre cas : bien qu'il soit possible de générer le descripteur d'une donnée qui soit la plus informative possible pour le modèle courant, ce descripteur ne serait pas interprétable par l'expert. Si l'on voulait générer des données qui le soient, il faudrait pouvoir synthétiser des paires de vidéos liées pour le problème à deux classes ou une vidéo issue d'un groupe de vidéos liées pour le problème à N classes. Dans l'état de nos connaissances, il n'existe pas d'approche qui en soit capable, même en utilisant des réseaux de neurones génératifs de type GAN. Afin de minimiser le nombre de sollicitations de l'expert, la stratégie au fil de l'eau semble la plus adaptée à notre contexte. Ainsi, l'oracle ne sera sollicité que pour une vidéo à la fois. Nous détaillons maintenant les descripteurs utilisés pour décrire les objets et l'arrière-plan des vidéos.

4.5.1 Caractéristiques utilisées pour décrire les vidéos

L'utilisation des histoires en tant que caractéristique pose un problème : d'une histoire à une autre, il peut y avoir différents nombres d'objets, chacun décrit par une apparence. Or, les descripteurs doivent être de taille fixe pour pouvoir les comparer. Nous choisissons

donc de compter la proportion de chaque catégorie d'objet sur différents intervalles de temps sans tenir compte de l'apparence. Nous choisissons également un nombre fini de catégories : « personne », « voiture », « camion », « moto », « vélo » et « autre » pour les objets qui n'entrent dans aucune des 5 premières catégories. Le descripteur contient les proportions des 6 catégories d'objets des tranches temporelles de longueur $0,1T$ (T étant la durée de la vidéo), avec un décalage de $0,05T$, ce qui représente 114 dimensions.

Outre ces caractéristiques issues de la détection des objets, nous considérons également des caractéristiques portant sur l'arrière-plan. Nous utilisons pour cela l'algorithme de segmentation sémantique décrit dans [Zhu 19] sur les modèles de décor des vidéos calculés par l'approche [Russell 06], comme présenté dans la section 2. Nous considérons alors différents découpages de l'image de la segmentation sémantique en rectangles de même taille selon les configurations suivantes :

— 3 tranches horizontales ;

— 3 tranches verticales ;

— un quadrillage régulier de 3×3 cellules ;

— l'image complète.

Nous caractérisons chacun de ces rectangles par les proportions de pixels appartenant à chacune des 20 classes sémantiques, ce qui représente un descripteur d'arrière-plan à 320 dimensions. Deux descripteurs d'arrière-plan $\mathbf{b}_1$ et $\mathbf{b}_2$ seront comparés à l'aide de l'indice de Jaccard, détaillé en annexe B, qui ici peut s'écrire :

$$\text{Jaccard}(\mathbf{b}_1, \mathbf{b}_2) = \sum_i \frac{\min(\mathbf{b}_1^i, \mathbf{b}_2^i)}{\max(\mathbf{b}_1^i, \mathbf{b}_2^i)}, \tag{4.6}$$

où les $\mathbf{b}^i$ représentent les proportions de pixels des différentes classes sémantiques dans les différentes configurations (tranches horizontales, tranches verticales, cellules et image complète). Ce descripteur est indépendant du descripteur des objets. Deux vidéos peuvent avoir des arrière-plans très différents tout en présentant du recouvrement dans leurs champs de vue, comme illustré par la figure 4.7. À l'inverse, lorsque les segmentations sémantiques de l'arrière-plan sont similaires, cela peut signifier que les caméras ont des points de vue et des orientations proches et présentent du recouvrement. Ainsi :

— il est difficile de déduire que deux vidéos présentent ou non du recouvrement à partir de segmentations sémantiques de l'arrière-plan différentes ;

— en revanche, des segmentations sémantiques similaires traduisent plus souvent un lien de recouvrement qu'une absence de recouvrement.

La suite de ce chapitre décrit les protocoles expérimentaux que nous avons mis en place pour la recherche des liens entre vidéos (classification à 2 classes) et la recherche des groupements de vidéos (classification à N classes). Nous comparons ensuite les résultats

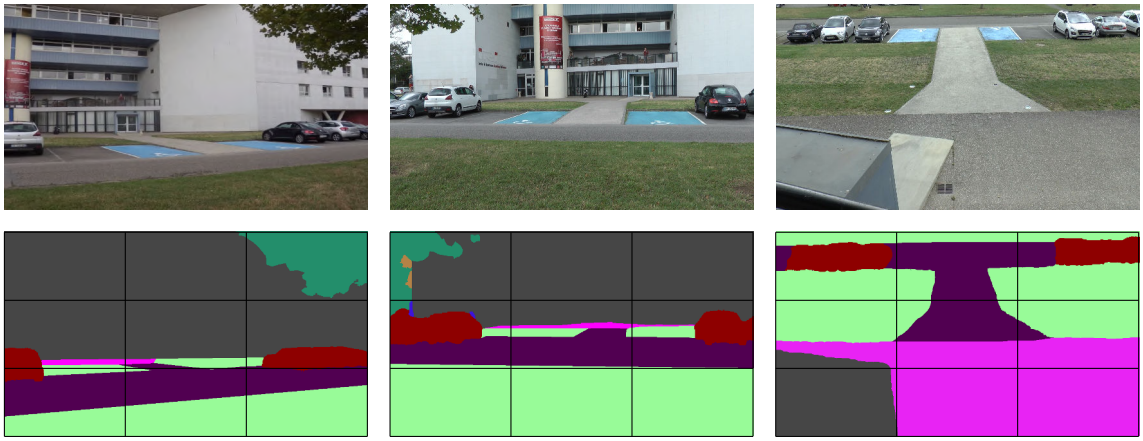


FIGURE 4.7 – Exemples de segmentations sémantiques de caméras avec recouvrement — En haut, 3 vues de notre jeu de données ToCaDa [Malon 18]. En bas, les segmentations sémantiques de référence correspondantes. Le quadrillage indique le découpage le plus fin réalisé, en cellules, et aide également à visualiser les tranches horizontales et verticales. Les deux premières caméras ont des points de vue et des orientations proches. Les proportions des classes sémantiques sont similaires entre les deux vues, en particulier le long des tranches horizontales et des tranches verticales. La troisième vue présente du recouvrement avec les deux autres mais leur fait face. Ses proportions de classes sémantiques par tranches et cellules présentent beaucoup moins de similarités avec les deux autres.

obtenus avec et sans utilisation de l'apprentissage actif. Ces expériences s'appuient sur des interfaces que nous avons développées pour l'occasion. Ces interfaces ont vocation à faire office de prototypes, aussi nous ne mettrons pas l'accent sur les problématiques qui leur sont propres : on ne cherchera pas, par exemple, à ce qu'elles soient prévisibles (le résultat d'une interaction est-il conforme à ce que l'utilisateur attendrait ?) ou contrôlable (lorsqu'un changement s'opère à l'écran, peut-on et sait-on facilement retrouver le contrôle ?).

4.5.2 Recherche des liens de recouvrement par apprentissage actif

Dans cette section, nous décrivons la méthode correspondant au cas de classification à 2 classes { liées, non liées } sur un ensemble de m vidéos. Plus précisément, dans le contexte d'une enquête, ces 2 classes sont relatives à une vidéo d'intérêt V donnée. Cette vidéo d'intérêt est généralement une vidéo dans laquelle on voit un délit ou une attaque être commis. On cherche donc à classer les $m - 1$ autres vidéos comme présentant du recouvrement avec la vidéo d'intérêt V. À chaque vidéo sont attribués deux descripteurs modélisant respectivement ses propriétés d'arrière-plan et d'objets détectés dans le temps, comme décrit dans la section 4.5.1. L'utilisateur est supposé être un expert du domaine : lorsqu'il est sollicité sur une paire de vidéos, il choisit la bonne classe avec une probabilité

p. Cette approximation permet, lorsque $p < 1$, de prendre en compte le fait que, même pour un humain, il n'est pas toujours facile d'affirmer que deux vidéos présentent du recouvrement ou non. Afin d'évaluer notre approche sans prendre en compte les erreurs potentielles de l'homme, nous nous placerons dans le cas idéal où $p = 1$.

Au départ, 3 seuils $\sigma_{pas\ lien} \leq \sigma_{hesitation} \leq \sigma_{lien}$ compris entre 0 et 1 sont initialisés. Pour chaque paire de vidéos (V_1, V_2) dont les descripteurs d'objets présents sont ($\mathbf{x}_1$, $\mathbf{x}_2$) et de descripteurs d'arrière-plan respectifs ($\mathbf{b}_1$, $\mathbf{b}_2$), l'algorithme agit selon la valeur de $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2 / (||\mathbf{x}_1|| \, ||\mathbf{x}_2||)$ selon le mécanisme décrit ci-après et illustré par la figure 4.8 :

— si $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \in]\sigma_{lien} ; 1]$, l'algorithme classe les vidéos comme étant liées dans le sens où elles présentent du recouvrement dans leurs champs de vue, sans solliciter l'oracle ;

— si $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \in]\sigma_{hesitation} ; \sigma_{lien}]$, l'algorithme s'attend à ce que les vidéos soient liées mais il « hésite » : si les descripteurs d'arrière-plan sont similaires, il classe les vidéos comme étant liées, sinon il sollicite l'oracle ;

— si $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \in]\sigma_{pas\ lien} ; \sigma_{hesitation}]$, l'algorithme sollicite l'oracle et s'attend à ce que les vidéos ne soient pas liées ;

— si $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \in [0 ; \sigma_{pas\ lien}]$, l'algorithme classe les vidéos comme étant non liées dans le sens où elles ne présentent pas de recouvrement dans leurs champs de vue, sans solliciter l'oracle .

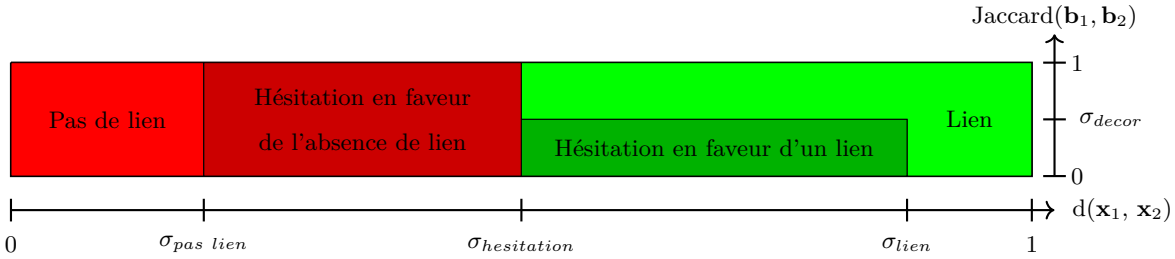


FIGURE 4.8 – Mécanisme de décision de notre algorithme de recherche de liens par apprentissage actif — Selon la similarité entre les descripteurs des vidéos $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2)$ et entre les descripteurs d'arrière-plans calculée par $\mathrm{Jaccard}(\mathbf{b}_1, \mathbf{b}_2)$, notre algorithme peut prendre une décision de lui-même ou solliciter l'oracle en cas d'hésitation, auquel cas il fait une supposition et adapte certains seuils en fonction de la concordance entre sa supposition et la réponse de l'oracle.

Lorsque $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \in]\sigma_{pas\ lien} ; \sigma_{lien}]$, l'algorithme « hésite » et sollicite l'oracle tout en penchant pour l'une ou l'autre classe selon que $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2)$ est inférieure ou supérieure à $\sigma_{hesitation}$. Dans le cas où il « hésite » et penche pour un lien entre les vidéos, il compare au préalable les descripteurs d'arrière-plan et classe les vidéos comme étant liées si $\mathrm{Jaccard}(\mathbf{b}_1, \mathbf{b}_2) > \sigma_{decor}$. En revanche, si $\mathrm{Jaccard}(\mathbf{b}_1, \mathbf{b}_2) \leq \sigma_{decor}$, l'oracle est sollicité.

Cette comparaison n'est pas faite dans le cas où l'algorithme penche pour une absence de recouvrement des champs de vue, car elle ne dissiperait pas l'hésitation. En effet, nous avons constaté que lorsque les arrière-plans sont dissemblables, cela ne présage en rien du recouvrement ou non des champs de vue des caméras. À l'inverse, lorsque les arrière-plans sont similaires, il est fréquent que les caméras aient leurs champs de vue en recouvrement et rare que ce ne soit pas le cas. La figure 4.9 présente les différents liens que nous avons constatés entre similarité d'arrière-plans et recouvrement des champs de vue des caméras.

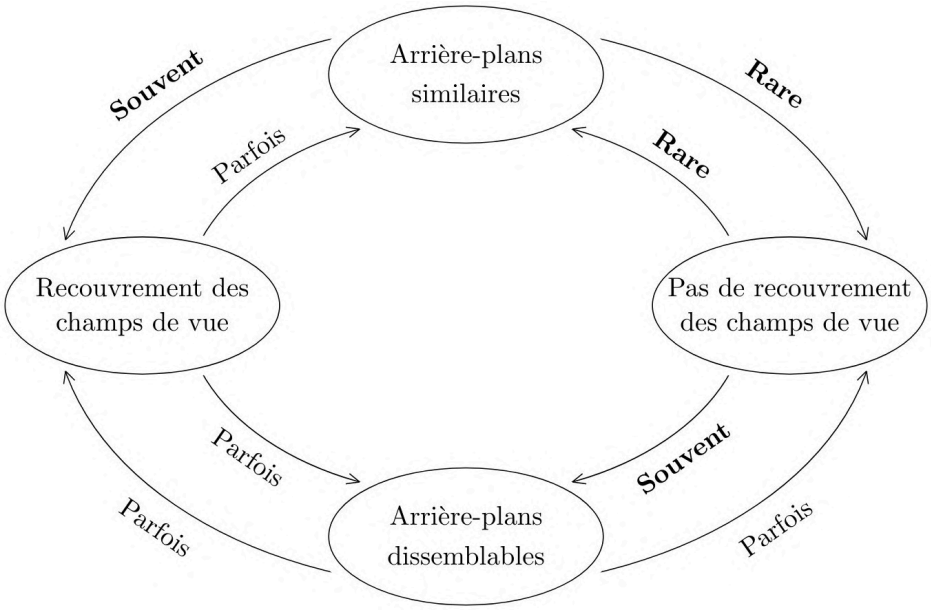


FIGURE 4.9 – Liens entre similarité d'arrière-plans et recouvrement des champs de vue — Nous mettons en gras les liens les plus discriminants pour la prise de décision.

Comme nous classons les vidéos relativement à la vidéo d'intérêt V, $\mathbf{x}_2$ et $\mathbf{b}_2$ sont respectivement le descripteur des objets présents dans V et le descripteur d'arrière-plan de V. Concernant la mise à jour des seuils, lorsque l'oracle prédit la classe à laquelle l'algorithme s'attendait, le seuil σ_{lien} se rapproche de $\sigma_{hesitation}$ à hauteur de $v\%$ de l'écart entre eux, v étant un paramètre de vitesse d'apprentissage proportionnel au nombre de vidéos non encore classées. Ainsi, l'algorithme prendra plus souvent la décision par lui-même de classer deux vidéos de faible distance comme étant liées. À l'inverse, le seuil $\sigma_{hesitation}$ se rapproche de σ_{lien} à hauteur de $v\%$ de leur écart de sorte que l'algorithme s'attende plus souvent à ce que deux vidéos soient non liées. Au fur et à mesure des prédictions et des sollicitations de l'oracle, les seuils s'affinent. La vitesse d'apprentissage v, proportionnelle au nombre de vidéos non encore classées, diminue progressivement. Une fois que la totalité des vidéos sont classées relativement à la vidéo d'intérêt V, l'algorithme prend fin. Nous décrivons maintenant l'interface associée à cette méthode.

4.5.3 Description de l'interface dans le cas à 2 classes

L'interface se compose de deux parties. La partie gauche est consacrée aux sollicitations de l'expert : lorsqu'une annotation est requise, c'est ici que s'affiche la vidéo concernée, ainsi que la vidéo d'intérêt V pour permettre à l'utilisateur de visualiser les deux simultanément. Plusieurs éléments sont présents dans la partie gauche de l'interface :

1. deux boutons « liées » et « non liées » qui permettent d'annoter la paire de vidéos dans la classe correspondante ;
2. une barre de lecture qui permet de mettre les deux vidéos à un même temps donné ;
3. un bouton de lecture qui permet de jouer les deux vidéos de façon synchronisée et qui dispose de 4 états (pause, lecture en vitesse normale, lecture en vitesse $\times$ 2, lecture en vitesse $\times$ 4) qui alternent à chaque clic.

En effet, n'avoir qu'une image de chaque vue peut suffire à statuer de l'absence de recouvrement des champs de vue lorsque l'arrière-plan présente de grandes différences, mais, en général, il est nécessaire de visualiser au moins une partie des deux vidéos pour répondre.

La partie droite donne un récapitulatif de l'état actuel des liens annotés et de ceux inférés. Les miniatures des vidéos et les liens annotés manuellement et appris par la machine y apparaissent. Les liens sont représentés par des segments liant la vidéo d'intérêt V, placée au centre, aux autres vidéos, placées en cercle autour. Les vidéos classées comme étant liées sont placées dans la partie droite du cercle et les segments représentant leurs liens avec la vidéo centrale sont en traits pleins tandis que les vidéos classées comme n'étant pas liées sont placées dans la partie gauche du cercle et les segments représentant l'absence de liens sont en pointillés. Enfin, la couleur de chaque segment indique s'il a été inféré automatiquement, auquel cas le segment apparaît en noir, ou si c'est l'oracle qui a donné la classe, auquel cas il apparaît en bleu. La disposition des miniatures est mise à jour de façon dynamique chaque fois qu'une vidéo est classée. La figure 4.10 illustre cette interface.

4.5.4 Apprentissage actif des groupements de vidéos liées

Dans cette section, nous décrivons la méthode correspondant au cas d'une classification à N classes $\{c_1, \ldots, c_N\}$ où chaque classe représente un ensemble de vidéos connexe, c'est-à-dire que l'on peut passer de n'importe quelle vidéo de la classe à n'importe quelle autre en se déplaçant seulement entre vidéos présentant du recouvrement. Si trois caméras C_1, C_2 et C_3 sont telles que :

— C_1 et C_2 ont du recouvrement dans leurs champs de vue ;

— C_2 et C_3 ont du recouvrement dans leurs champs de vue ;

— C_1 et C_3 n'ont pas de recouvrement dans leurs champs de vue ;

alors C_1, C_2 et C_3 appartiennent à la même classe.

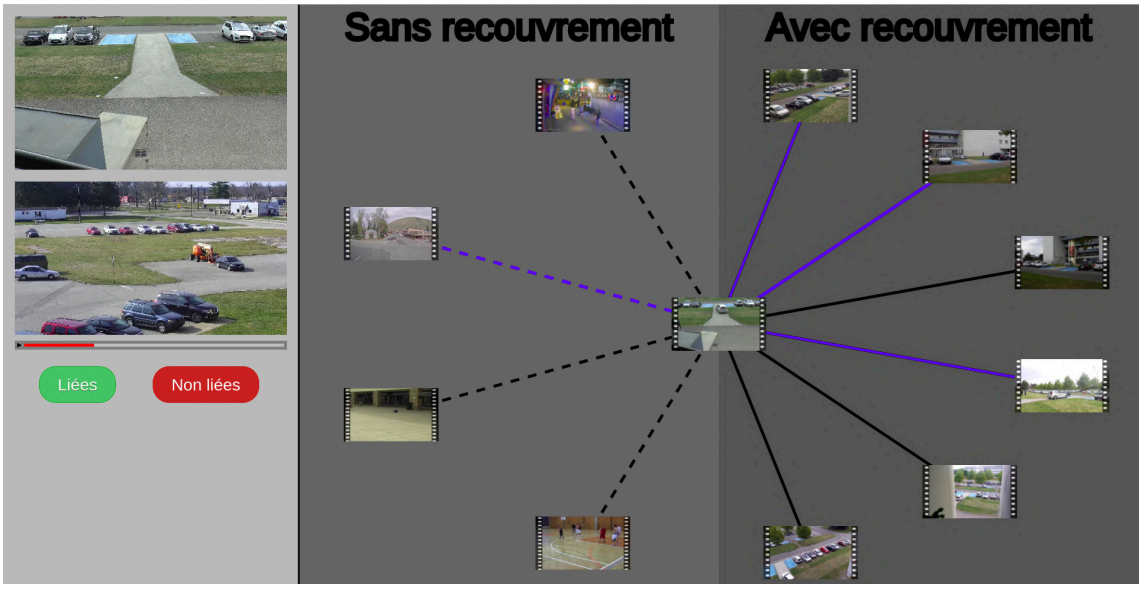


FIGURE 4.10 – Interface proposée d'apprentissage actif dans le cas à 2 classes — Les miniatures des vidéos classées sont disposées en cercle autour de la vidéo d'interêt. Les segments en trait plein représentent un lien de recouvrement des champs tandis que ceux en pointillés représentent une absence de lien. La couleur de chaque segment indique s'il a été inféré automatiquement, auquel cas il apparaît en noir, ou si c'est l'oracle qui a donné la classe, auquel cas le segment apparaît en bleu.

Dans cet exemple, si C_1 et C_3 se retrouvent dans deux classes disjointes au cours de la formation des groupes, il faut pouvoir combiner ces deux classes par la suite. Pour cela, l'idée sera d'utiliser le fait que C_2 est similaire à la fois à C_1 et à C_3 qui sont pourtant de deux classes différentes et de solliciter l'oracle en lui donnant la possibilité de combiner les classes ou d'assigner C_3 à l'une des classes.

Initialement, la valeur du nombre de classes N n'est pas connue. Il n'y a cette fois pas de vidéo d'intérêt qui se distingue des autres. À nouveau, on considérera que l'utilisateur, expert du domaine, choisit la bonne classe avec une probabilité p, prise égale à 1 dans nos expériences, lorsqu'il est sollicité.

Comme pour le cas à 2 classes, nous définissons 3 seuils $\sigma_{pas\ lien} \leq \sigma_{hesitation} \leq \sigma_{lien}$ compris entre 0 et 1. Pour chaque vidéo V_i qui n'est pas encore classée, de descripteur $\mathbf{x}_i$, on calcule les distances $\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$ entre $\mathbf{x}_i$ et les descripteurs $\mathbf{x}_j$ des vidéos V_j déjà classées. En fonction des valeurs des deux mesures de similarité les plus grandes $d' \geq d''$ obtenues avec les vidéos V' et V'', différents cas de figure peuvent se produire. Nous les détaillons dans le tableau 4.1. Les actions sont les mêmes que dans le cas à 2 classes lorsque d' et d'' ne tombent pas dans le même intervalle (configurations 2, 3, 4, 6, 7 et 9). Lorsque d' et d''sont toutes deux inférieures ou égales à $\sigma_{pas\ lien}$ (configuration 1), une nouvelle classe est créée et initialisée avec V. Lorsque d' et d'' sont dans le même intervalle et sont supérieures à $\sigma_{pas\ lien}$ (configurations 5, 8 et 10), l'oracle est sollicité et peut choisir de

créer une nouvelle classe, d'attribuer V à une classe déjà existante, ou encore de combiner les classes de V' et de V'' et d'y ajouter V. Dans le cas où V' et V'' appartiennent à la même classe, il n'est pas proposé de combiner les classes. Les différents seuils et la vitesse d'apprentissage v sont mis à jour de façon similaire au cas à 2 classes selon que l'oracle classe une vidéo pour laquelle il est sollicité dans la même classe ou non que celle attendue par l'algorithme.

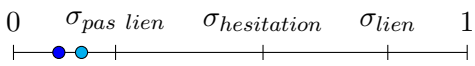
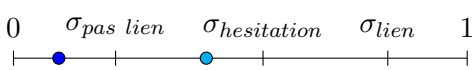
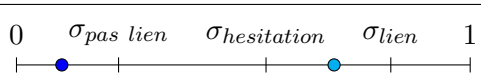
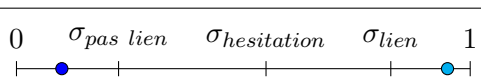
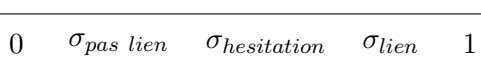
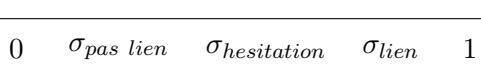
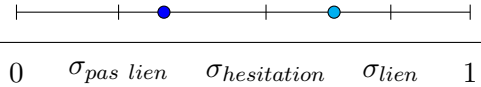
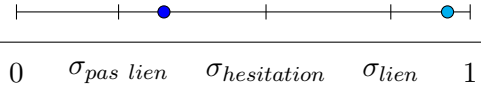
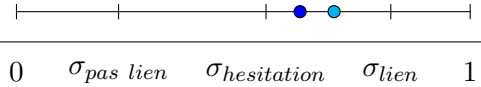
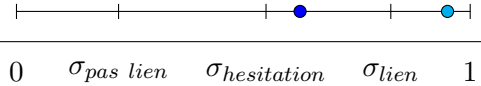
	Configuration	Action
1)		Créer une nouvelle classe.
2)		Solliciter l'oracle pour classer V.
3)		Classer V comme V' si $\mathbf{b}$ et $\mathbf{b}'$ sont similaires. Solliciter l'oracle pour classer V sinon.
4)		Classer V comme V'.
5)		Solliciter l'oracle pour classer V ou pour combiner V avec les classes de V' et de V''.
6)		Classer V comme V' si $\mathbf{b}$ et $\mathbf{b}'$ sont similaires. Solliciter l'oracle pour classer V sinon.
7)		Classer V comme V'.
8)		Solliciter l'oracle pour classer V ou pour combiner V avec les classes de V' et de V''.
9)		Classer V comme V'.
10)		Solliciter l'oracle pour classer V ou pour combiner V avec les classes de V' et de V''.

Tableau 4.1 – Configurations possibles dans le cas de classification à N classes — Les points cyan et bleus correspondent à des valeurs possibles de d' et d''. À chaque configuration d'appartenance de d' et d'' aux intervalles $[0\ ;\ \sigma_{pas\ lien}]$, $]\sigma_{pas\ lien}\ ;\ \sigma_{hesitation}]$, $]\sigma_{hesitation}\ ;\ \sigma_{lien}]$ et $]\sigma_{lien}\ ;\ 1]$, l'action effectuée par l'algorithme est indiquée.

4.5.5 Description de l'interface dans le cas à N classes

L'interface se présente, de façon similaire à celle du cas à 2 classes, en deux parties. Cette fois, nous présentons d'abord la partie droite de l'interface : on y voit, pour chaque classe, les miniatures des vidéos de la classe disposées en cercle. Les centres des N cercles

correspondant aux N classes sont eux-mêmes disposés en cercle. Il n'y a pas de segments entre les vidéos ; au sein d'une classe, il existe une suite de vidéos qui permet de passer de n'importe quelle vue à n'importe quelle autre en se déplaçant entre paires de caméras aux champs de vue se recouvrant.

La partie gauche, présentant les sollicitations, n'affiche qu'une vidéo à la fois. La barre de lecture et le bouton de lecture fonctionnent comme dans le cas précédent. En revanche, le choix de la classe d'un individu $\mathbf{x}_i$ pour lequel l'expert est sollicité se fait en cliquant dans la partie droite de l'interface :

— ou bien dans l'un des cercles déjà existants pour l'ajouter à la classe ;

— ou bien en dehors de tous les cercles pour créer une nouvelle classe et l'initialiser avec l'individu $\mathbf{x}_i$;

— ou bien, pour fusionner deux classes et y mettre l'individu, par deux clics droits successifs sur les deux classes.

La disposition des différentes miniatures correspondant aux vidéos est mise à jour de façon dynamique comme présenté dans l'annexe C.2 chaque fois qu'une vidéo est classée, soit par le programme, soit par l'oracle. Une capture d'écran de l'interface est donnée sur la figure 4.11.

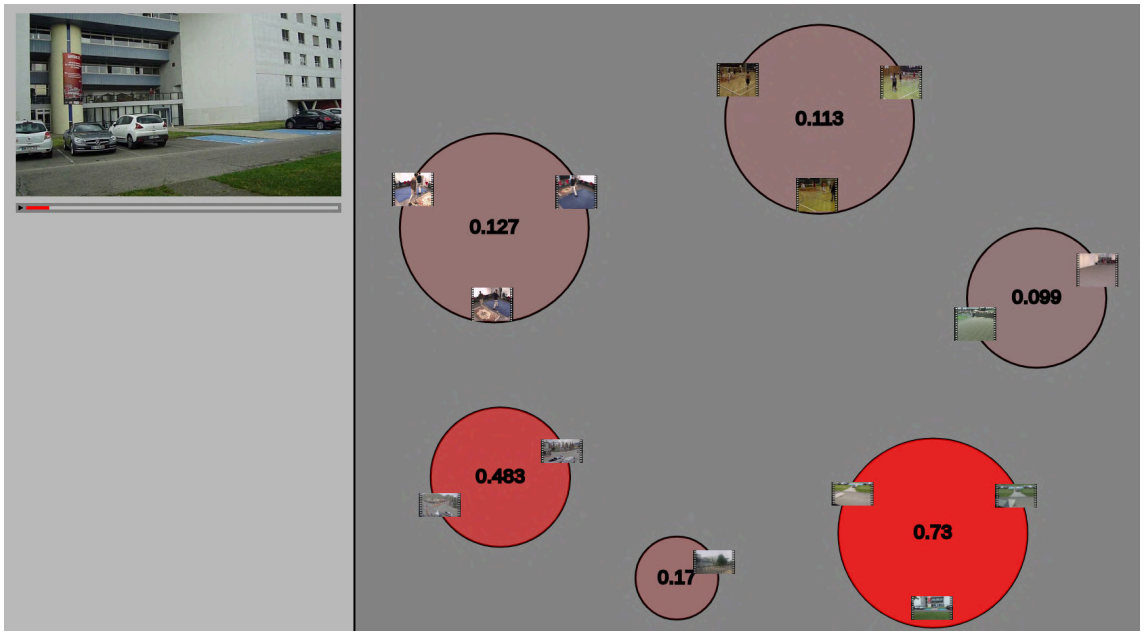


FIGURE 4.11 – Interface proposée d'apprentissage actif des groupements de vidéos en recouvrement — Pour une donnée non étiquetée, présentée à gauche, le modèle estime la probabilité d'appartenance à chacune des classes qui correspondent à des groupements de vidéos dont les champs de vue se recouvrent. Ils sont représentées par des cercles contenant les miniatures des vidéos et colorés d'un rouge d'autant plus vif que le modèle donne une forte probabilité, affichée au centre du cercle, à cette classe.

4.5.6 Évaluation de l'apprentissage actif de liens

Nous commençons par rappeler les différents paramètres utilisés dans nos approches :

— le nombre de vidéos considérées m ;

— la vitesse d'apprentissage v ;

— la probabilité p que l'oracle classe correctement une vidéo ;

— le seuil de similarité des arrière-plans σ_{decor} ;

— les seuils associés aux stratégies de sollicitation initialisés respectivement à $\sigma_{lien} = 1$, $\sigma_{pas\ lien} = 0$ et $\sigma_{hesitation} = 0{,}5$.

Notre jeu de données comprend au total 63 vidéos issues des mêmes jeux de données que dans le chapitre précédent. Nous considérons 5 valeurs différentes pour le nombre de vidéos m : 5, 10, 20, 40 et 63 vidéos. Nous distinguons les expériences du cas à 2 classes de celles du cas à N classes.

Dans le cas à 2 classes, l'une des m vidéos est sélectionnée aléatoirement pour être la vidéo d'intérêt et l'expérience consiste à classer les $m-1$ autres vidéos comme présentant ou non du recouvrement. Pour chacune des tailles d'ensembles de m vidéos, l'expérience est répétée en tirant aléatoirement à 100 reprises un sous-ensemble de m vidéos parmi les 63 dont nous disposons. Les 2 classes sont ici connues dès le début de l'expérience et nous déterminons, à chaque expérience, le taux de vidéos correctement classées, souvent appelé **précision**. Nous calculons ensuite la **précision moyenne** obtenue à partir des 100 expériences pour chaque taille d'ensemble de m vidéos. Nous souhaitons également connaître le nombre moyen de sollicitations de l'oracle. En termes de comparaisons avec des approches existantes, le nombre de classes étant connu à l'avance, nous comparons ces résultats avec ceux obtenus par l'algorithme des k-moyennes avec $k = 2$.

Les deux paramètres principaux dont nous voulons déterminer l'impact sont le nombre de vidéos m et le seuil de similarité des arrière-plans σ_{decor}, qui peut permettre de classer une vidéo sans solliciter l'oracle lorsque l'algorithme hésite en faveur d'un lien de recouvrement. Quatre valeurs sont testées pour ce seuil. Notons que $\sigma_{decor} = 1$ revient à toujours solliciter l'oracle en cas d'hésitation. La vitesse d'apprentissage a été fixée empirquement à $v = 0{,}1$. Concernant la probabilité de bonne classification de l'oracle, nous choisissons de la prendre égale à $p = 1$ afin de mesurer les performances de notre approche dans le cas d'un oracle qui ne se trompe jamais. En pratique, la probabilité d'erreur n'est pas uniforme, elle dépend de l'oracle, de la vidéo qu'on lui demande de classer et des vidéos déjà classées. La figure 4.12 présente les résultats obtenus dans le cas où $p = 1$.

Concernant l'impact de σ_{decor}, une faible valeur implique qu'une similarité faible des arrière-plans suffit à classer une vidéo avec la vidéo d'intérêt sans demander à l'oracle, ce qui se traduit par de nombreuses erreurs et moins de sollicitations. À l'opposé, les meilleures valeurs de précision sont obtenues lorsque l'oracle est systématiquement sollicité

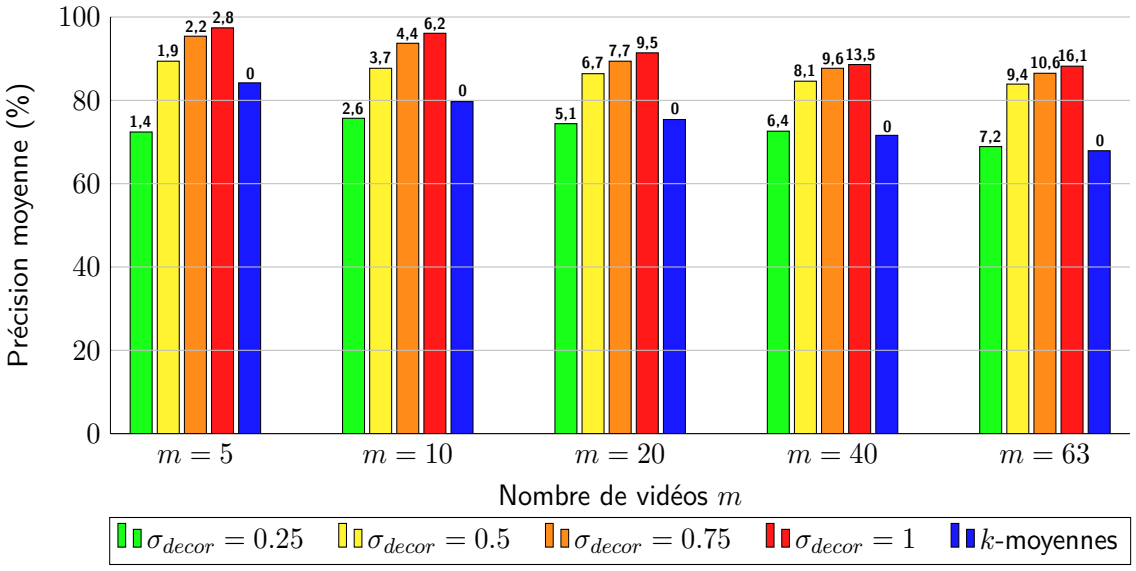


FIGURE 4.12 – Précision moyenne et nombre moyen de sollicitations de l'oracle de notre approche dans le cas à 2 classes – Pour différentes valeurs du nombre m de vidéos considérées, nous mesurons la précision moyenne obtenue sur 100 répétitions de notre expérience dans le cas à 2 classes pour différentes valeurs du seuil de similarité de l'arrière-plan σ_{decor}. Nous mesurons également la précision moyenne obtenue par l'algorithme des k-moyennes avec $k = 2$. Nous indiquons au-dessus de chaque barre le nombre moyen de sollicitations de l'oracle. La vitesse d'apprentissage et la probabilité que l'oracle choisisse la bonne classe lorsqu'il est sollicité sont fixées respectivement à $v = 0{,}1$ et $p = 1$.

en cas d'hésitation, mais le nombre de sollicitations moyen représente un pourcentage significatif du nombre total de vidéos : près d'une vidéo sur deux lorsque $m = 20$ ou d'une vidéo sur quatre lorsque $m = 63$. Les deux valeurs de σ_{decor} intermédiaires offrent un compromis intéressant entre la précision obtenue et le nombre de sollicitations de l'oracle. Nous choisissons de fixer $\sigma_{decor} = 0{,}5$. À part lorsque $\sigma_{decor} = 0$, notre méthode donne une meilleure précision que l'algorithme des k-moyennes.

Le nombre m de vidéos tirées aléatoirement parmi les 63 vidéos a essentiellement un impact sur le nombre de sollicitations de l'oracle. Pour de petits ensembles de vidéos ($m = 5$, $m = 10$ ou $m = 20$), l'oracle est très souvent sollicité proportionnellement au nombre de vidéos à classer. En effet, l'algorithme dispose alors de peu d'exemples, ce qui le pousse généralement à hésiter. À mesure que m augmente, le nombre de sollicitations croît de moins en moins vite : il peut mieux de mieux ajuster ses seuils σ_{lien}, $\sigma_{pas\ lien}$ et $\sigma_{hesitation}$ et ainsi moins solliciter l'oracle. L'impact de m sur la précision est plus limité : à mesure que le nombre de vidéos augmente, elle diminue légèrement car l'algorithme choisit plus souvent par lui-même, risquant potentiellement de faire des erreurs. Nous nuançons tout de même ce constat en rappelant que l'oracle peut lui aussi commettre des erreurs.

Dans le cas à N classes, nous tirons de la même façon à 100 reprises m vidéos aléatoirement parmi toutes celles disponibles pour différentes valeurs de m. La différence principale

se situe au niveau de la mesure effectuée : les classes n'étant pas connues à l'avance et se créant au fur et à mesure de chaque expérience, il est difficile d'estimer le taux de bonne classification. Il n'y a pas forcément de correspondance entre les groupements formés et la classification de référence attendue : il est possible que l'on n'ait pas trouvé le bon nombre de groupements (c'est-à-dire de classes) à l'issue d'une expérience. Afin de définir la précision moyenne dans ce cas-là, nous mesurons le nombre minimal moyen de changements de classe nécessaires puis calculons le pourcentage moyen de vidéos nécessitant un changement de classe, soit entre classes existantes soit d'une classe existante vers une nouvelle classe. Nous employons alors le terme de précision moyenne pour désigner le pourcentage moyen de vidéos ne nécessitant pas un changement de classe, ce qui se rapproche de la définition classique. De nouveau nous mesurons les performances de notre méthode pour différentes valeurs du seuil de similarité de l'arrière-plan σ_{decor}. Nous mesurons également les performances obtenues par l'algorithme *mean-shift*. Plusieurs valeurs du paramètre de bande passante ont été testées. Expérimentalement, les meilleurs résultats sont généralement obtenus pour des bandes passantes proches de 0,3. Nous choisissons donc cette valeur pour le paramètre de bande passante r. La figure 4.13 présente les résultats obtenus.

En moyenne, les précisions moyennes obtenues sont inférieures de 10% à celles du cas à 2 classes. Le nombre moyen de sollicitations est, quant à lui, de 10% à 15% plus élevé que dans le cas à 2 classes, et l'utilisation de l'arrière-plan permet moins souvent de classer une vidéo sans solliciter l'oracle. Cela s'explique par les stratégies adoptées dans les différentes configurations présentées à la figure 4.1 : seules les configurations 3 et 6 font appel à la similarité d'arrière-plan, tandis que dans le cas de la configuration 8, l'oracle est aussitôt consulté sans que l'arrière-plan ne soit considéré. À nouveau, les meilleurs résultats sont obtenus pour $\sigma_{decor} = 1$, c'est-à-dire lorsque l'on consulte systématiquement l'oracle sans utiliser la similarité des décors. Nous remarquons toutefois qu'à mesure que le nombre de vidéos augmente ($m = 40$ et $m = 63$), des résultats similaires sont obtenus pour σ_{decor} =0,5 et demandent 33% à 35% moins de sollicitations de l'oracle. Enfin, *mean-shift* donne des résultats comparables aux nôtres pour un faible nombre de vidéos ($m = 5$ ou $m = 10$) mais de moins bons au-delà.

4.6 Conclusion

Dans ce chapitre, nous avons présenté les différents types d'apprentissage couramment distingués : supervisé, non supervisé et semi-supervisé. Concernant l'apprentissage semi-supervisé, nous avons détaillé l'apprentissage actif qui présente l'avantage de pouvoir solliciter l'expert pour l'aider dans sa tâche de classification, notamment sur les données que son modèle a du mal à étiqueter. Plusieurs approches utilisent l'apprentissage actif pour classer des données multimédias, mais peu le font sur des vidéos à partir de leur contenu. L'approche qui nous semble la plus proche de la nôtre est celle décrite dans [Khoshrou 14].

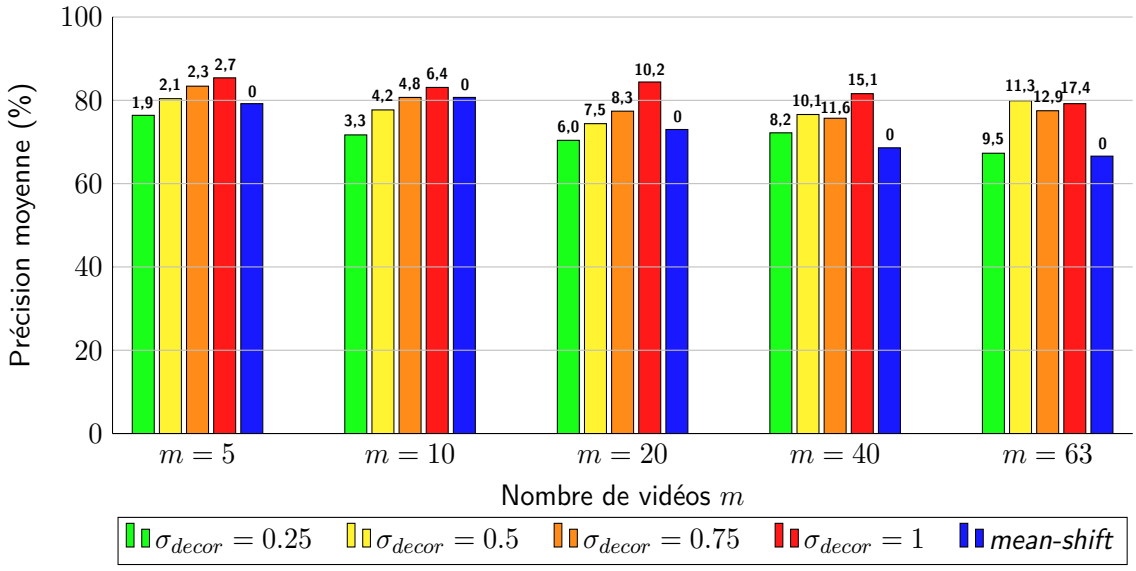


FIGURE 4.13 – Précision moyenne et nombre moyen de sollicitations de l'oracle de notre approche dans le cas à N classes – Pour différentes valeurs du nombre m de vidéos considérées, nous mesurons la précision moyenne obtenue sur 100 répétitions de notre expérience pour différentes valeurs du seuil de similarité de l'arrière-plan σ_{decor}. Nous mesurons également la précision moyenne obtenue par l'algorithme *mean-shift* avec le paramètre de bande passante pris égal à $r = 0{,}3$. Nous indiquons au-dessus de chaque barre le nombre moyen de sollicitations de l'oracle. La vitesse d'apprentissage et la probabilité que l'oracle choisisse la bonne classe sont fixées respectivement à $v = 0{,}1$ et $p = 1$.

Nous nous en inspirons dans notre contribution en permettant à l'oracle de créer dynamiquement de nouvelles classes lorsqu'il est sollicité dans le cas à N classes. Nous aonvs mesuré la précision de la classification dans le cas à 2 classes et dans le cas à N classes et parvenons à classer correctement plus de 80% des vidéos d'un ensemble de 63 vidéos en sollicitant l'oracle à une dizaine de reprises. Nous avons considéré que l'oracle ne commet aucune erreur de classement lorsqu'il est sollicité. En pratique, ce n'est pas le cas et nous envisageons de mettre en place une étude utilisateur pour mesurer l'impact des erreurs potentielles de l'oracle sur la classification. Dans le chapitre suivant, nous cherchons à permettre à l'utilisateur de naviguer d'une vidéo à une autre en formulant des requêtes.

Chapitre 5

Visualisation et navigation interactive entre vidéos

Contenu

5.1 Introduction

Dans le contexte d'une enquête, l'automatisation de la recherche de liens traitée dans les chapitres 2 et 4 a pour finalité d'accélérer le travail des opérateurs. Toutefois, la façon dont leur sont présentés ces liens est tout aussi importante que la validité de ces liens. Les opérateurs disposent généralement d'une interface sur laquelle s'affichent des données et des résultats issus de traitements sur ces données (dans notre cas, les liens). La façon dont sont disposés ces éléments individuellement, mais aussi les uns par rapport aux autres, joue un rôle essentiel dans la facilitation du travail des enquêteurs. Nous touchons là au problème de la **visualisation** des données. Le nombre de données total étant généralement trop élevé pour pouvoir toutes les afficher à la fois, il est nécessaire d'effectuer une sélection. Même en imaginant que l'on dispose d'un très grand écran, la perception humaine ne parvient à traiter simultanément qu'un nombre limité de données. Selon les auteurs de [Barthel 17], pour des images, ce nombre se situe entre 20 et 50.

Ce problème de la visualisation va souvent de paire avec celui de la **navigation** : par quels mécanismes permet-on aux opérateurs de modifier la sélection des données affichées ? Dans ce chapitre, nous présentons des travaux existants qui mêlent visualisation et navigation, avant de proposer une contribution adaptée à notre contexte et permettant une navigation par le biais de requêtes trajectographiques.

5.2 Approches existantes de visualisation et de navigation

Le choix des données à afficher est particulièrement important lorsque l'on manipule de grands ensembles de données. À titre d'exemple introductif, quand on effectue une recherche de certains mots-clés via un moteur de recherche, plusieurs millions de résultats de recherche sont trouvés. Ne pouvant pas tous les afficher simultanément, le moteur de recherche les trie généralement en fonction de certains critères de pertinence et le navigateur Web n'affiche que les plus pertinents sur la page, généralement par ordre décroissant de pertinence. L'utilisateur a alors la possibilité de sélectionner le résultat de la page qui correspond le plus à ce qu'il recherche, ou à voir les résultats des pages suivantes. Dans cet exemple, l'interaction se situe dans la possibilité d'entrer les mots-clés de la recherche et de choisir le résultat dans la liste. Dans la suite, nous présentons des approches traitant de la visualisation et de la navigation sur des ensembles d'images ou des vidéos. Nous récapitulerons les caractéristiques principales de chacunes des approches dans le tableau 5.1 situé à la page 103. Nous distinguons trois échelles d'approches :

— à l'échelle de plusieurs vidéos, celles qui cherchent à présenter de nombreuses images ou vidéos différentes à l'écran ;

— à l'échelle d'une vidéo, les approches qui cherchent à présenter un résumé aussi complet que possible des instants d'intérêt de la vidéo ;

— à l'échelle de portions d'une vidéo, celles qui cherchent à réduire le temps nécessaire au visionnage de l'intégralité de la vidéo.

5.2.1 Visualisation et navigation dans des données nombreuses

Nous commençons par présenter les travaux existants concernant la visualisation et la navigation parmi des nombreuses images ou vidéos. Les auteurs de [Barthel 17] affichent des miniatures issues d'un ensemble de 4 millions d'images dans un plan vu en perspective, comme illustré par la figure 5.1. Elles sont disposées de sorte que des miniatures soient d'autant plus proches qu'elles se ressemblent visuellement. À partir d'une recherche par mot-clé, l'utilisateur est placé dans ce tapissage de miniatures au niveau de celles qui correspondent au mot-clé. Des recommandations sont proposées à l'utilisateur pour préciser sa recherche et le rediriger vers un sous-ensemble des miniatures. Par exemple, s'il a cherché « félin », il peut lui être proposé de préciser entre « chat » ou « tigre ». Les auteurs appliquent leur approche sur des images uniquement.

FIGURE 5.1 – L'interface de navigation de [Barthel 17]. À partir du mot-clé « chat », un tapissage d'images correspondantes est affiché. L'utilisateur peut y naviguer ou choisir parmi des sous-catégories plus précises qui lui sont proposées. Figure issue de [Barthel 17].

L'interface proposée dans [Schaefer 11] permet une navigation multi-échelle dans une multitude d'images. À l'échelle la plus fine, les images sont placées dans une grille hexagonale, ou structure en nid d'abeilles, où chaque hexagone contient une image, comme illustré sur la figure 5.2. Les images sont placées les unes par rapport aux autres en fonc-

tion de leur similarité de couleurs : plus les couleurs de deux images sont similaires, plus ces images sont proches. Chaque fois que l'on passe à une échelle supérieure, on trace un pavage hexagonal où chaque hexagone couvre plusieurs images de l'échelle inférieure et est représenté par l'une des images couvertes. La navigation se fait par des clics sur un hexagone pour passer à l'échelle inférieure, ainsi que par des raccourcis dans un menu à gauche dans l'interface qui permet de revenir à la résolution supérieure ou à la résolution initiale, la plus grossière. Enfin, l'utilisateur peut indiquer qu'une image l'intéresse pour qu'elle soit affichée en permanence dans une barre en bas de l'interface. Il peut alors revenir sur cette image à souhait en cliquant directement dans cette barre.

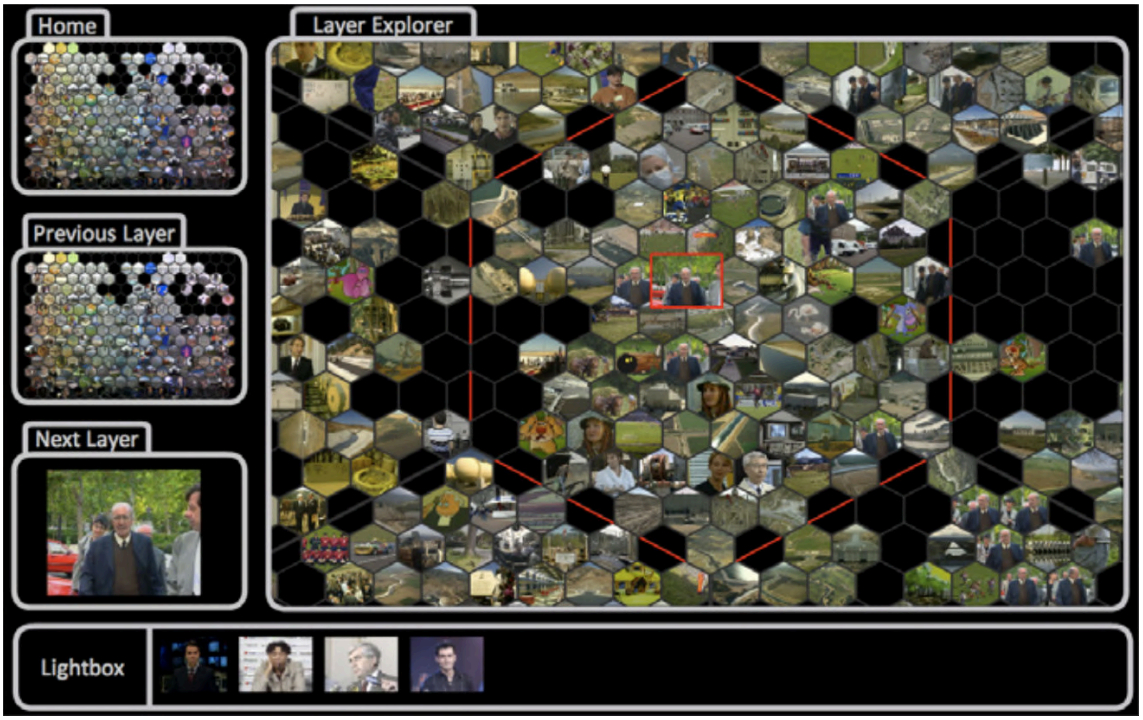


FIGURE 5.2 – L'interface de navigation proposée dans [Schaefer 11] — « *Layer Explorer* » correspond à la couche actuelle dans laquelle l'utilisateur peut naviguer, « *Home* » correspond à la résolution initiale qui est la plus grossière, « *Previous Layer* » permet de revenir à un niveau d'échelle en arrière et « *Next Layer* » permet d'aller à un niveau d'échelle plus fin. Enfin « *Lightbox* » contient des vidéos que l'utilisateur a sélectionnées comme étant d'intérêt pour y revenir facilement en cliquant dessus. Figure issue de [Schaefer 11].

Les graphes, composés de nœuds et d'arêtes matérialisant des relations entre certains nœuds, peuvent aussi être utilisés voire même rendus explicitement visibles dans les systèmes de visualisation et de navigation. L'interface disponible à l'adresse `http://www.wineandcheesemap.com/`, et illustrée sur la figure 5.3, en est un exemple. Au départ, une vue d'ensemble des différents vins et fromages, représentés par des nœuds de différentes couleurs, est présentée à l'utilisateur qui peut librement zoomer et se déplacer par translation. Les arêtes, quant à elles, représentent les mariages classiques entre différents fromages

et vins. Lors d'un clic sur l'un des nœuds, l'interface se réarrange pour centrer le nœud cliqué et seuls les nœuds qui sont en relation avec le nœud sélectionné restent affichés et se disposent autour de celui-ci selon un cercle. La navigation s'effectue alors de proche en proche en cliquant à chaque fois sur un des nœuds du cercle. Il est possible de revenir à la disposition initiale où tous les nœuds sont affichés en cliquant en dehors des nœuds.

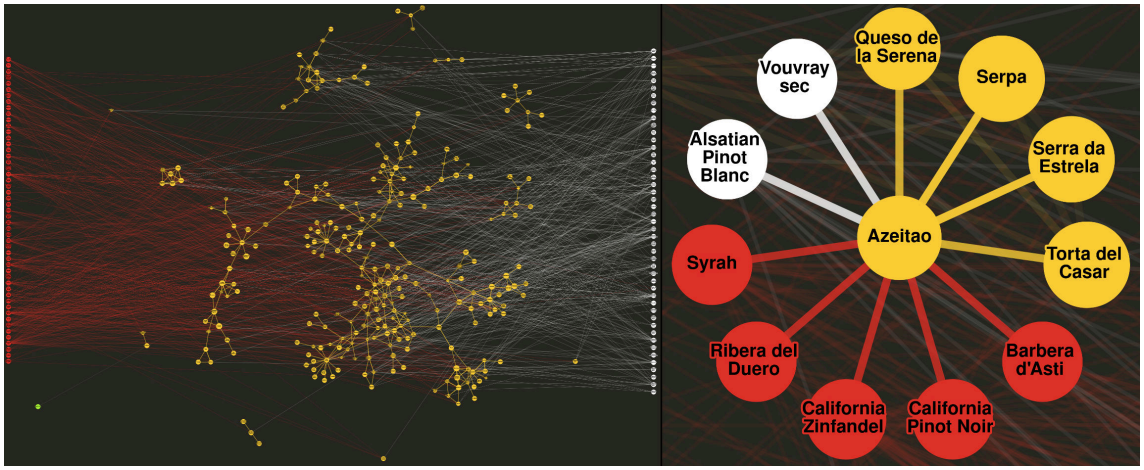


FIGURE 5.3 – Navigation dans le graphe des fromages et des vins — À gauche, une vue d'ensemble du graphe composé de nœuds représentant des fromages (en jaune) ou des vins (en rouge et blanc) et d'arêtes symbolisant des accords de saveurs. À droite, après un clic sur un nœud, le graphe se réarrange et place le nœud cliqué — ici un fromage — au centre et les nœuds qui lui sont liés autour.

Enfin, chaque année se tient le *Video Browser Showdown*, un challenge au cours duquel différents outils sont testés sur des tâches de recherche dans de grandes collections de vidéos. De nombreuses interfaces innovantes sont proposées chaque année. L'outil qui a remporté le challenge en 2019, `vitrivr`, est présenté en détails dans [Rossetto 19] et illustré sur la figure 5.4. L'utilisateur a la possibilité de formuler différents types de requêtes : il peut sélectionner une image stockée sur son ordinateur et demander à rechercher des fichiers vidéos dans lesquels on retrouve des teintes de couleurs proches, il peut faire des recherches par mot-clé, ou encore dessiner une esquisse sémantique de l'arrière-plan de la scène. Quel que soit le mode de recherche, l'outil affiche les vidéos qui correspondent le plus, classées par ordre décroissant de pertinence. L'utilisateur peut alors sélectionner une de ces miniatures et en faire sa nouvelle requête. Les participants ont eu accès aux vidéos pendant plusieurs mois avant le challenge et disposaient ainsi de beaucoup de temps pour les annoter manuellement. Nous citons également [Barthel 15], sans en illustrer l'interface car elle combine un affichage similaire à [Barthel 17] et une fonctionnalité de dessin d'esquisse comme dans [Rossetto 19].

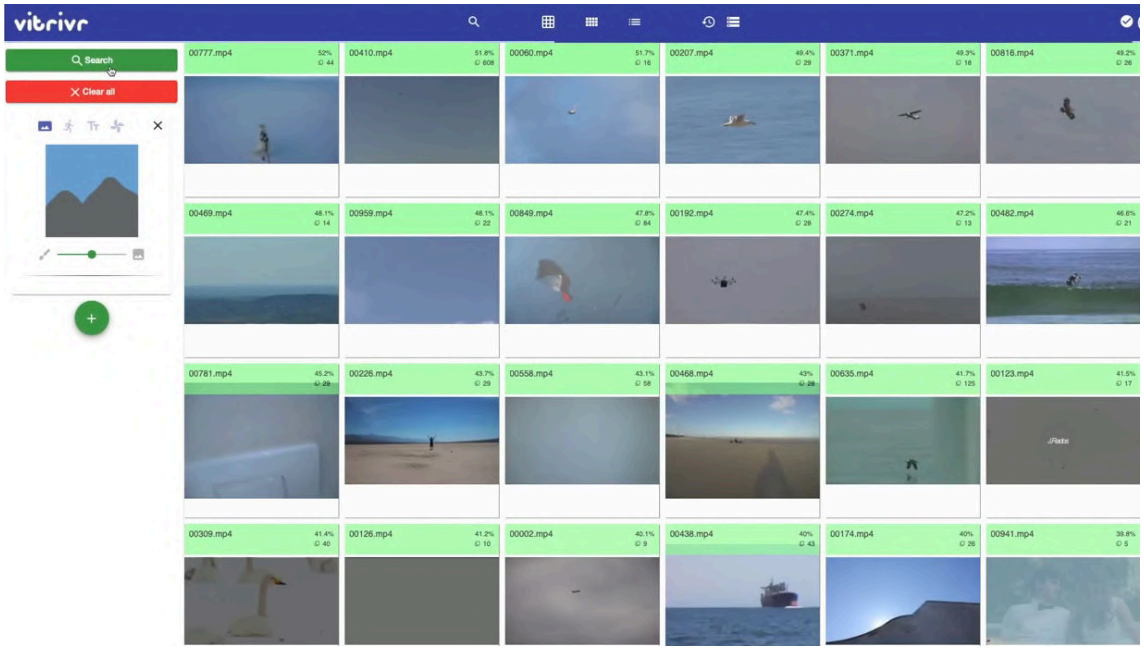


FIGURE 5.4 – L'interface de navigation proposée dans [Rossetto 19]. Sur la gauche, une esquisse très simple a été faite par un utilisateur. L'interface affiche alors des miniatures de vidéos dont les couleurs correspondent à l'esquisse. Figure issue de [Rossetto 19].

5.2.2 Affichages simultanés

Dans le contexte de la visualisation de vidéos, les approches cherchent généralement à condenser l'information contenue dans le temps pour retranscrire un maximum de l'information contenue dans les vidéos en un minimum de temps, et ainsi éviter à l'utilisateur de devoir les regarder intégralement. Cela passe par un affichage simultané d'informations contenues à différents instants de la vidéo ou des vidéos. Différents critères peuvent être utilisés pour choisir les éléments à afficher simultanément et leur disposition les uns par rapports aux autres.

Pour permettre la visualisation de plus de contenu d'intérêt en simultané, [Kang 06] combinent les régions de différentes vidéos en une même vidéo. Ils associent à chaque pixel un score d'intérêt en fonction du mouvement qu'il présente dans le temps et regroupent les pixels de haut score en régions d'intérêt. Ils identifient ainsi spatialement et temporellement les portions de vidéos qui présentent le plus d'activité. Les régions d'intérêt de portions de vidéos sont ensuite placées côte à côte sur les faces d'une structure cubique, comme illustré par la figure 5.5. Les aspects d'interface et d'interaction ne sont pas traités. Nous pouvons imaginer faire tourner la structure cubique pour choisir la face que l'on souhaite visionner et pouvoir cliquer sur une portion de vidéo pour être redirigé vers la vidéo originale.

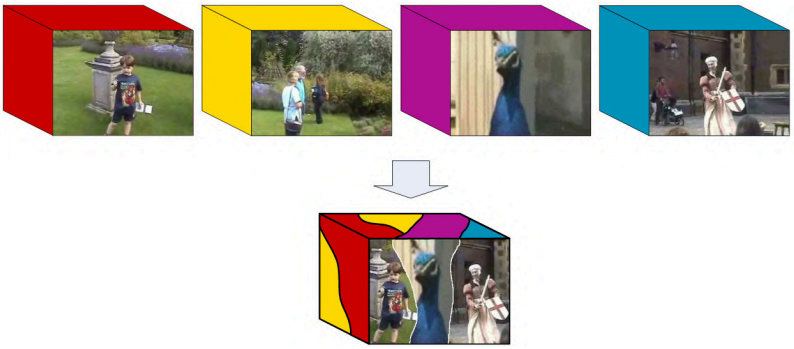


FIGURE 5.5 – Combinaison de vidéos [Kang 06] — À partir de différentes vidéos, les portions temporelles et spatiales présentant le plus de mouvement sont identifiées et placées côte à côte. Les différentes faces du cube obtenu présentent différents arrangements des portions de vidéos. Figure issue de [Kang 06].

Dans le même esprit, les auteurs de [Pritch 08] proposent d'afficher simultanément les activités s'étant déroulées à différents instants et en différents endroits d'une même vidéo, comme illustré par la figure 5.6. Similairement à [Kang 06], ils identifient spatialement et temporellement les portions de la vidéo où il y a du mouvement, puis produisent une vidéo qui résume l'activité en affichant simultanément des portions de vidéo qui ne se recouvrent pas spatialement. L'utilisateur peut formuler une requête spécifiant la période de temps dont il souhaite un résumé et la durée de ce résumé, telle que « obtenir le résumé des activités principales de la vidéo en une minute ». Dans cet exemple, seules les portions où il y a le plus de mouvement sur une durée totale d'une minute sont gardées. Cette approche de résumé de vidéo est intéressante dans un contexte de surveillance car, sur une partie significative du temps, aucune activité n'a lieu dans le champ de vue de la plupart des caméras de surveillance.

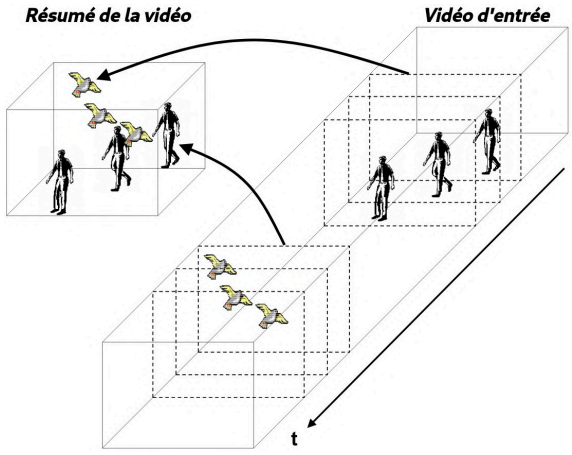


FIGURE 5.6 – Résumé de vidéo par affichages simultanés [Pritch 08] — Après avoir détecté les régions d'activité au cours du temps dans la vidéo d'entrée, ces dernières sont combinées et affichées simultanément pour proposer une vidéo condensée. Figure issue de [Pritch 08].

Enfin, les auteurs de [Caspi 06] traitent le cas des vidéos où une personne est en mouvement mais que sa position dans l'image change peu. C'est le cas par exemple pour une personne au loin qui s'approche de la caméra, ou encore pour des mouvements d'étirement. Dans ces cas-là, l'affichage superposé d'images clé de la personne pose problème car les différentes images s'occultent les unes les autres. Les auteurs proposent alors d'afficher côte à côte les images clés, comme illustré sur la figure 5.7.

FIGURE 5.7 – Affichage d'images clés côte à côte — Pour les cas où un objet ou une personne d'intérêt se déplace peu, les auteurs de [Caspi 06] proposent d'afficher côte à côte des images clés de la vidéo qui rendent compte de son mouvement. Dans cet exemple, la danse de la petite fille est bien retranscrite. Figure issue de [Caspi 06].

5.2.3 Résumés globaux

Les méthodes d'affichage simultané permettent de condenser l'information contenue dans une vidéo sur une durée plus courte. D'autres approches cherchent à permettre de visualiser simultanément les informations contenues dans l'intégralité d'une vidéo en proposant un résumé des activités d'intérêt qui y ont lieu. Ici aussi, les critères utilisés pour identifier les informations à faire figurer dans le résumé peuvent être de différentes natures.

Les auteurs de [Nguyen 12] proposent d'utiliser un parallélépipède où les deux premières dimensions correspondent à celles d'une image de la vidéo et où la troisième dimension représente le temps, comme illustré sur la figure 5.8. Un ensemble d'images clés est extrait de la vidéo, soit uniformément en choisissant une image sur 20, soit en associant à chaque image une probabilité d'être une image clé qui est proportionnelle à la quantité de mouvement dans cette image. Le cube est découpé spatialement en voxels $v_{x,y,t}$ où x et y correspondent aux coordonnées pixelliques dans l'image et t au temps dans la vidéo, plus spécifiquement l'indice d'image clé. En termes d'interactions, l'utilisateur peut zoomer ou dézoomer sur le cube, lui faire subir une rotation ou une translation. Il peut rendre transparents les voxels où il n'y a pas de mouvement. Enfin, il peut allonger la dimension temporelle du cube pour voir plus en détails les images clés.

FIGURE 5.8 – Représentation d'une vidéo sous forme de parallélépipède — Les différentes images de la vidéo sont placées sur une troisième dimension qui correspond au temps. Seules certaines images clés sont gardées. Figure issue de [Nguyen 12].

D'autres techniques cherchent à produire une synthèse d'activité de la vidéo basée sur une ligne de temps marquée par les évènements. Dans le chapitre 4, nous avons évoqué l'article [Khoshrou 14], illustré par la figure 4.6, dont l'objectif est de fournir un résumé de la présence d'objets dans différentes caméras au cours du temps sous la forme de barres de temps indiquant l'intervalle de temps de présence pour chaque objet.

En plus de l'information de présence, une autre caractéristique des objets souvent exploitée est la trajectoire. Dans un contexte d'enquête, il n'est pas rare de rechercher un objet ayant suivi une trajectoire particulière. Plusieurs méthodes de résumés globaux ont été adaptées à ce contexte. Les auteurs de [Höferlin 11] proposent une interface dans laquelle sont affichées les trajectoires les plus suivies dans une vidéo. Les trajectoires similaires sont regroupées au sein d'une famille de trajectoires. Pour chaque famille de trajectoires, une ligne de temps est affichée et des marques indiquent tous les instants où un objet a suivi une des trajectoires, comme schématisé sur la figure 5.9. L'utilisateur peut préciser les familles de trajectoires qui l'intéressent pour ne visualiser que celles-ci. Il peut également cliquer sur une marque dans la ligne de temps pour visionner la séquence du passage de l'objet correspondant.

Dans leurs travaux, les auteurs de [Meghdadi 13] affichent les positions successives des personnes ainsi que leurs trajectoires dans un repère 3D composé des dimensions spatiales de l'image et de la dimension temporelle, comme illustré sur la figure 5.10. Ils proposent également une vue d'ensemble des évènements, c'est-à-dire des instants où des piétons sont présents, avec la possibilité de spécifier un intervalle de temps et de sélectionner un évènement pour visualiser le suivi de l'objet correspondant.

FIGURE 5.9 – Résumé des trajectoires suivies dans une vidéo — Pour une vidéo donnée, trois familles de trajectoires suivies par des objets apparaissant au cours de la vidéo sont représentées, respectivement en rouge, en vert et en bleu. En bas, des lignes de temps correspondant à chaque famille de trajectoires indiquent les instants où un objet a suivi une trajectoire de la famille. À gauche de chaque ligne de temps est indiqué le nombre d'objets total ayant suivi une trajectoire de la famille, et à droite est indiqué un score de diversité des trajectoires de la famille. Figure issue de [Höferlin 11].

Enfin, dans leurs travaux, les auteurs de [Höferlin 09] proposent différents types de filtres pour rechercher les objets ayant suivi certains types de trajectoires dans une vidéo, comme illustré par la figure 5.11. Un premier type de filtre permet de sélectionner une région de l'image : les objets dont la trajectoire sort de la région sélectionnée sont alors exclus des résultats de recherche. Un second type de filtre concerne l'orientation des trajectoires. Il permet de ne conserver que les objets dont la trajectoire est orientée selon certains angles que l'on spécifie. Ce type de filtre est efficace pour retrouver des objets dont la trajectoire est une ligne droite, mais ne permet pas d'identifier une trajectoire plus complexe où l'objet change d'orientation au cours de la trajectoire.

Après avoir passé en revue les différentes méthodes de visualisation et de navigation dans des contenus d'images ou de vidéos, récapitulées dans le tableau 5.1, nous pouvons maintenant nous demander lesquelles semblent les plus pertinentes dans notre contexte d'enquête.

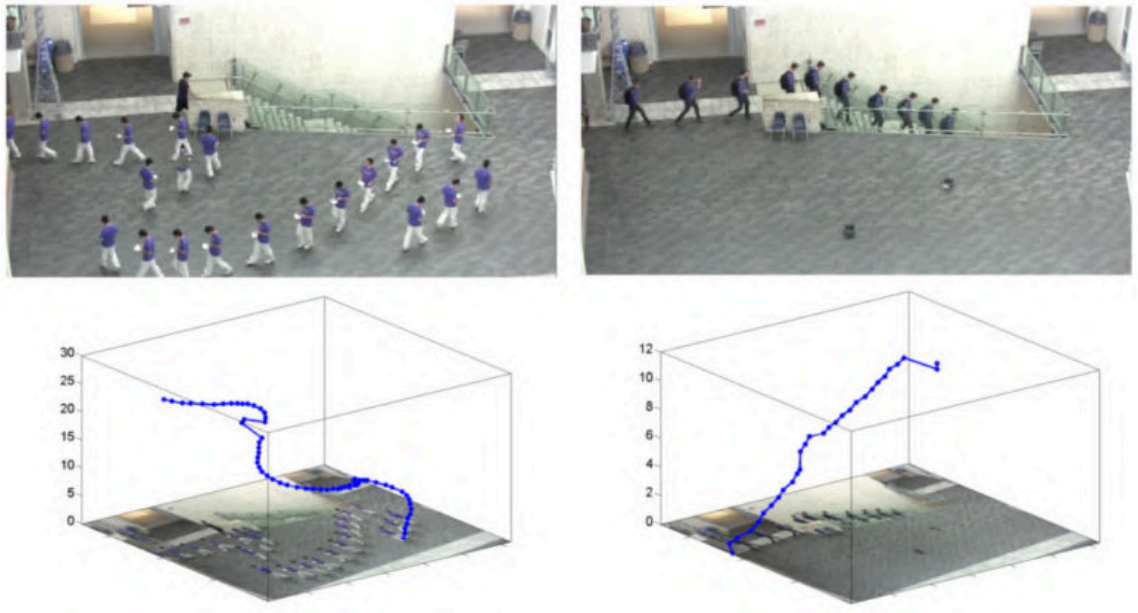


FIGURE 5.10 – Affichage des positions successives et trajectoires 3D — Deux exemples d'évènements correspondant aux suivis de deux piétons. Les trajectoires 3D sont également affichées, la troisième dimension étant le temps. Figure issue de [Meghdadi 13].

5.3 Aide à la navigation dans une collection de vidéos par reformulation de trajectoires

Dans cette partie, nous présentons une troisième et dernière contribution. Il s'agit d'une approche de classement d'une collection de vidéos avec recouvrement. À partir d'une trajectoire requête tracée dans l'une des vidéos, nous classons les autres vidéos, présentant d'autres points de vue, en plaçant en tête celles qui permettent de voir au mieux cette trajectoire. Notre approche estime une carte de correspondance entre régions de différentes vidéos en se basant sur la corrélation linéaire de leur taux d'occupation au cours du temps. À nouveau, nous exploitons l'idée que deux régions de deux vidéos différentes se correspondent d'autant mieux qu'elles sont systématiquement simultanément occupées ou non occupées. Nous utilisons ensuite les correspondances entre régions pour estimer la trajectoire reformulée dans chacune des autres vidéos puis les classons en fonction de la visibilité qu'elles en offrent.

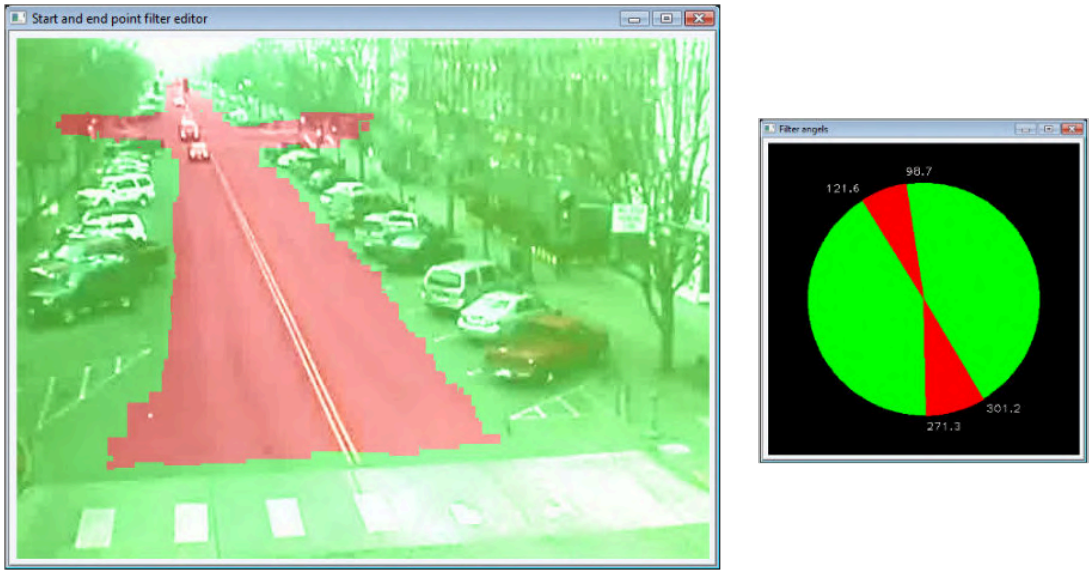


FIGURE 5.11 – Recherche d'objets filtrée par trajectoires — À gauche, une contrainte de position est illustrée : elle permet de n'obtenir que les objets dont la trajectoire est contenue dans la région rouge, délimitée par l'utilisateur. À droite, une contrainte d'orientation : elle permet de n'obtenir que les objets dont la trajectoire est orientée selon des angles compris dans les régions rouges, sélectionnés par l'utilisateur. Figure issue de [Höferlin 09].

5.3.1 Motivation de nos choix

Dans le domaine de la vidéosurveillance, les enquêteurs ont généralement des dizaines de caméras à traiter dont certaines pouvant présenter du recouvrement dans leurs champs de vue. À ce sujet, nous avons eu l'occasion d'échanger avec des enquêteurs impliqués dans le projet VICTORIA. Les tâches qu'ils sont régulièrement amenés à effectuer comprennent notamment :

1. trouver un véhicule qui a suivi une trajectoire donnée ;
2. trouver une autre vidéo dans laquelle apparaît avec plus de détails une personne qui a été aperçue dans une première vidéo ;
3. trouver toutes les personnes qui sont entrées ou sorties d'un bâtiment ;
4. trouver tous les instants où il y a eu de l'activité dans une région d'intérêt.

Pour naviguer entre les vidéos, les enquêteurs nous ont dit préférer formuler manuellement des requêtes sur des éléments d'intérêt puis pouvoir choisir parmi une liste classée de vidéos qui correspondent à leur requête, plutôt que d'être redirigés automatiquement d'une vidéo à l'autre.

Prenons l'exemple d'une attaque à la bombe dans un bâtiment. Après l'évènement, les enquêteurs reçoivent de grandes quantités de vidéos issues de caméras de surveillance ou filmées par des témoins. Partant d'une vidéo dans laquelle le bâtiment est visible, ils peuvent vouloir trouver toutes les personnes qui sont entrées dans le bâtiment ou en sont

Méthode	Type de données	Visualisation	Interactions
[Barthel 17]	Ensemble d'images	Sous-ensemble d'images similaires	Translation, zoom, recherche par mot-clé
[Schaefer 11]	Ensemble d'images	Sous-ensemble d'images similaires	Translation, changement d'échelle
[Rossetto 19]	Ensemble de vidéos	Miniatures correspondant à une requête	Recherche par mot-clé, par couleur, par image, esquisse de miniature
[Barthel 15]	Ensemble de vidéos	Miniatures correspondant à une requête	Esquisse de miniature
[Kang 06]	Ensemble de vidéos	Résumé par combinaison de régions des vidéos	/
[Pritch 08]	Vidéo individuelle	Résumé par superposition de régions	Spécification de la durée du résumé
[Caspi 06]	Vidéo individuelle	Affichage d'images clés côte à côte	/
[Nguyen 12]	Vidéo individuelle	Affichage 3D d'images clés	Rotation, zoom
[Khoshrou 14]	Ensemble de vidéos	Résumé des objets par barres de temps	Clic à un instant d'une barre de temps
[Höferlin 11]	Vidéo individuelle	Résumé des trajectoires par barres de temps	Clic à un instant d'une barre de temps
[Meghdadi 13]	Ensemble de vidéos	Résumé des trajectoires et affichage d'images clés	Clic à un instant d'une barre de temps
[Höferlin 09]	Vidéo individuelle	Séquences correspondant à une requête	Recherche de trajectoires et d'orientations
[Malon 19b]	Ensemble de vidéos	Vidéo correspondant à une requête	Requête trajectographique

Tableau 5.1 – Tableau récapitulatif des méthodes de visualisation et de navigation – La ligne en jaune correspond à notre contribution.

sorties aux alentours du moment de l'explosion. Parmi le grand nombre de vidéos dont ils disposent, celle dont ils partent n'est pas nécessairement celle qui capture le mieux l'entrée du bâtiment, que ce soit en raison d'éventuelles occultations, des conditions d'éclairage ou de l'angle de la caméra. Il leur serait donc pratique de pouvoir tracer une trajectoire requête dans cette vue, par exemple une flèche pointant vers l'entrée, et d'obtenir une liste, triée par ordre de pertinence, des vues dans lesquelles la trajectoire correspondante à la requête est plus étirée et offre donc une meilleure visibilité. Pour cette raison, nous avons choisi de proposer un mécanisme de navigation entre vidéos où les interactions se font par des trajectoires tracées à la main. Les travaux les plus proches que nous connaissons sont ceux décrits dans [Carlier 17] qui permettent à l'utilisateur de sélectionner une région d'intérêt dans une vidéo parmi une collection de vidéos présentant du recouvrement, et

d'être redirigé vers celle où les objets contenus dans la région désignée occupent la plus grande proportion de l'image. Cependant, ils supposent que les paramètres de calibrage des caméras sont connus. Dans un cas général, il peut être difficile d'avoir une estimation fiable de ces paramètres. Ainsi, dans ce travail, nous ne chercherons pas à les calculer.

5.3.2 Vue d'ensemble de la méthode proposée

Notre contribution principale consiste à proposer une nouvelle approche pour reformuler une trajectoire tracée dans une vue en ses trajectoires correspondantes dans d'autres vues en recouvrement pour proposer un classement de ces vues en fonction de la visibilité que chacune offre de la trajectoire reformulée. Cette façon de naviguer par le biais de requêtes trajectographiques est plutôt originale et nous a été inspirée d'une discussion avec des enquêteurs partenaires du projet VICTORIA.

Nous commençons par introduire les fonctions d'activité pour estimer les cartes de correspondance entre paires de vidéos. Puis, lorsqu'un utilisateur trace une requête trajectographique, nous proposons un score de reformulation pour trouver la trajectoire correspondante dans chacune des autres vidéos en recouvrement. Enfin, nous définissons un score de visibilité pour classer les vidéos en considérant la longeur de la trajectoire projetée dans cette caméra. En effet, nous supposons que plus une trajectoire apparaît comme étendue dans une vue, plus cette trajectoire a une chance d'être détaillée, c'est-à-dire de fournir un point de vue plus informatif où on peut espérer obtenir de nouveaux détails sur cette trajectoire et sur l'action relative à cette trajectoire.

Nous supposons que chaque vidéo a été filmée depuis un point de vue statique, et que l'ensemble des vidéos est temporellement synchronisé : les vidéos commencent donc au même instant et ont la même durée. Nous ne disposons d'aucun paramètre de calibrage des caméras et nous ne chercherons pas à les estimer.

Les étapes successives de notre approche sont décrites par la figure 5.12. À partir d'une collection de vidéos, en (a), nous détectons les objets d'intérêt et leur assignons une catégorie, comme illustré en (b) où des boîtes englobantes rouges correspondent aux véhicules et une boîte englobante bleue correspond à un piéton. Cette étape peut être effectuée en utilisant l'un des algorithmes de détection présentés dans le chapitre 2, par exemple YOLO [Redmon 16]. Ensuite, en (c), nous décomposons chaque vue en cellules et associons à chacune une fonction d'activité par catégorie d'objet. La fonction d'activité est définie comme étant le taux d'occupation de la cellule par un objet, d'une catégorie donnée, au cours du temps. Puis, pour chaque paire de vidéos, en (d), nous assignons, à chaque cellule de la première vidéo de la paire, une carte de correspondance indiquant la correspondance avec les cellules de la deuxième vidéo. Cette mise en correspondance est effectuée en fonction de la corrélation entre les fonctions d'activité (deux cellules sont mises en valeur dans la vue du haut et la carte de correspondance apparaît sur la vue du dessous).

Un utilisateur peut tracer manuellement une trajectoire requête dans l'une des vues, en (e). À partir de la carte de correspondance dans les autres vues de chaque cellule traversée par la requête, nous obtenons une trajectoire reformulée en choisissant la séquence des cellules correspondantes qui offre le meilleur score de reformulation. Ce score maximise le rapport entre, d'une part, la correspondance des cellules de la trajectoire reformulée avec celles de la trajectoire d'origine et, d'autre part, la distance entre les cellules consécutives de la trajectoire reformulée. Un classement des vidéos en fonction de ce score de reformulation et de la visibilité qu'elles offrent de la trajectoire reformulée est finalement obtenu en (f).

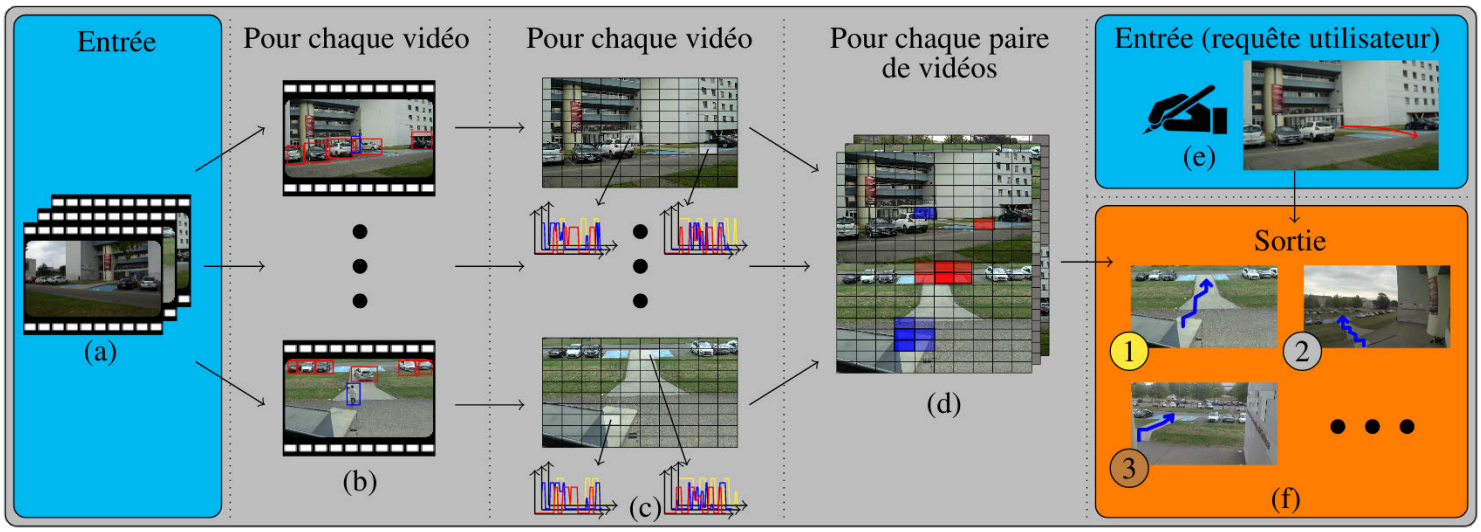


FIGURE 5.12 – Vue d'ensemble de l'approche : (a) En entrée, une collection de vidéos, (b) détection des objets et de leurs catégories, (c) calcul des fonctions d'activité, (d) calcul des cartes de correspondance, (e) requête trajectographique d'un utilisateur, (f) classement des vidéos en fonction de leur score de visibilité.

5.3.3 Cartes de correspondance

Dans cette section, nous définissons la notion de fonction d'activité et explicitons notre façon de calculer la carte de correspondance entre chaque région d'une vue et les régions qui lui correspondent dans une autre vue.

Nous découpons chaque vidéo V en $N \times N$ cellules $c_{\mathbf{i}}^V$ selon une grille régulière, où $\mathbf{i} = [i, j]^{\mathrm{T}}$ où $(i, j) \in [1, N]^2$ sont respectivement l'indice de la colonne et l'indice de la ligne de la cellule $c_{\mathbf{i}}^V$. L'impact du choix de la valeur de N est étudié dans la section 5.3.7. Le traitement des images d'une vidéo sera ainsi fait par cellule et non par pixel, afin de réduire les temps de calculs, d'une part, et d'avoir une information plus riche que celle contenue dans un seul pixel, d'autre part.

Pour une vidéo V, notons $d_\omega^V(t)$ l'ensemble des objets de la catégorie ω détectés au temps t. La fonction d'activité de la catégorie ω dans la vue V, notée $a_{\mathbf{i}}^{V,\omega}$, est le taux de recouvrement de la cellule $c_{\mathbf{i}}^V$ par des détections d_ω^V des objets de la même catégorie ω au

cours du temps (voir figure 5.13) :

$$a_{\mathbf{i}}^{V,\omega}(t) = \frac{|c_{\mathbf{i}}^{V} \cap d_{\omega}^{V}(t)|}{|c_{\mathbf{i}}^{V}|} \tag{5.1}$$

où $|\cdot|$ désigne le nombre de pixels. On notera qu'une cellule est associée à une fonction d'activité par catégorie d'objets (voir figure 5.12.(c)).

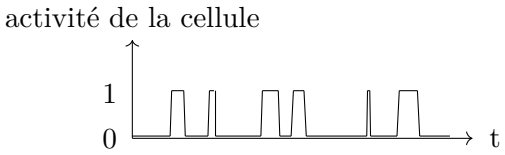


FIGURE 5.13 – Fonction d'activité d'une cellule pour une catégorie donnée : nous calculons la proportion de la cellule qui est occupée par un objet de la catégorie au cours du temps.

Soient V_1 et V_2 deux vidéos temporellement synchronisées présentant du recouvrement dans leurs champs de vue. Deux cellules de deux vidéos différentes $c_{\mathbf{i}}^{V_1}$ et $c_{\mathbf{i}'}^{V_2}$ ont de fortes chances de se correspondre si elles présentent simultanément et systématiquement de l'activité aux mêmes temps, c'est-à-dire si leurs fonctions d'activité sont fortement corrélées positivement.

Nous définissons ainsi le taux de correspondance entre deux cellules pour la catégorie ω par :

$$\mathcal{C}_{\omega}(c_{\mathbf{i}}^{V_1}, c_{\mathbf{i}'}^{V_2}) = \max(0, \mathrm{corr}(a_{\mathbf{i}}^{V_1,\omega}, a_{\mathbf{i}'}^{V_2,\omega})) \tag{5.2}$$

où corr est la corrélation linéaire entre deux variables aléatoires. Pour chaque cellule $c_{\mathbf{i}}^{V_1,\omega}$, nous définissons aussi une carte de correspondance $\mathcal{C}_{\omega}(c_{\mathbf{i}}^{V_1,\omega}, V_2)$ qui contient les scores de corrélations entre $c_{\mathbf{i}}^{V_1,\omega}$ et l'ensemble des cellules de V_2. La figure 5.12.(d) illustre les cartes de correspondance : une paire de vues est affichée et deux cellules sont mises en avant dans la vue du haut en bleu et en rouge, respectivement pour les catégories « piéton » et « véhicule ». La vue du bas montre les cartes de correspondance de ces deux cellules : en bleu, des cellules qui ont été occupées par un objet de la catégorie « piéton » lorsque la cellule en bleu de la vue du haut l'était simultanément, et de même en rouge pour la catégorie « voiture ».

Bien qu'il soit très improbable que deux cellules de deux vidéos différentes, et qui ne se correspondent pas, présentent systématiquement la même activité, cela peut se produire lorsque la durée des vidéos est courte. Si l'on dispose d'heures de vidéos synchronisées, ce qui est généralement le cas avec des caméras de surveillance, les régions qui se correspondent peuvent être d'autant mieux apprises.

D'autre part, la distinction des catégories permet d'éviter de faire correspondre une

cellule d'une vue qui pourrait être couverte par des objets de différentes tailles à de très nombreuses cellules dans une autre vue, comme illustré par la figure 5.14.

FIGURE 5.14 – Ambiguité de profondeur entre cellules. Deux vues avec recouvrement du jeu de données ToCaDa [Malon 18]. À gauche : une cellule est colorée en rouge. À droite : les cartes de corrélation correspondant à la cellule dans une autre vue pour les catégories « piéton » (en bleu) et « moto » (en jaune). Comme la cellule en rouge peut être couverte à la fois par les pieds d'une personne marchant à l'arrière plan derrière les voitures garées et par le haut d'une moto passant au premier plan, les cartes de correspondance couvrent deux groupes disjoints de cellules.

5.3.4 Reformulation de trajectoire

Dans cette section, nous présentons notre méthode de reformulation de trajectoire qui s'appuie sur les cartes de correspondance. Une trajectoire est définie par une ligne brisée, c'est-à-dire une succession de segments connectés. Pour une trajectoire requête donnée, nous noterons $\mathcal{S}$ la séquence de M cellules non consécutivement identiques $(c_{\mathbf{i}_1}, ..., c_{\mathbf{i}_M})$ qui sont traversées par les segments formant la trajectoire requête, voir figure 5.15.

FIGURE 5.15 – Exemple de trajectoire requête et de cellules traversées – en rouge, une trajectoire requête dessinée dans une vue par l'utilisateur. Les cellules traversées par cette requête sont mises en évidence en rose. C'est à partir des cartes de correspondance entre ces cellules et une autre vue que nous reformulons la trajectoire.

Pour trouver la trajectoire qui correspond à une trajectoire requête, dans une autre vue, nous voulons trouver les indices $(\mathbf{i'}_1, ..., \mathbf{i'}_M)$ des cellules correspondantes. Nous obtenons la trajectoire reformulée dans V_2 en joignant les centres des cellules de la séquence de cellules $(c_{\mathbf{i'}_1}, ..., c_{\mathbf{i'}_M})$. Pour trouver les indices, nous maximisons le rapport entre, d'une part, les corrélations des cellules traversées par la requête et les cellules traversées par la trajectoire reformulée et, d'autre part, un terme assurant la continuité de la trajectoire reformulée. Dans ce but, nous pénalisons les successions de cellules dans la trajectoire reformulée qui ne sont pas adjacentes. Le **score de reformulation** R d'une séquence de cellules $\mathcal{S}$ entre deux vues V_1 et V_2 est ainsi donné par l'expression :

$$R = \underset{(\mathbf{i'}_1,...,\mathbf{i'}_M)}{\operatorname{argmax}} \frac{\frac{1}{M}\sum_{k=1}^{M} \mathcal{C}_\omega(c_{\mathbf{i}_k}^{V_1}, c_{\mathbf{i'}_k}^{V_2})}{1 + \sum_{k=1}^{M-1} \max(0, ||\mathbf{i'}_{k+1} - \mathbf{i'}_k|| - 1)}. \tag{5.3}$$

Comme attendu, le numérateur favorise des cellules de V_2 qui ont une bonne correspondance avec les cellules traversées dans V_1. Le dénominateur pénalise des cellules consécutives qui ne sont pas adjacentes ou confondues dans la trajectoire reformulée. Les résultats de reformulation de trajectoires sont présentés dans la section 5.3.7.

5.3.5 Sélection et classement des vidéos

Dans cette section, nous exposons notre méthode de sélection et de classement des vidéos en fonction de la visibilité qu'elles offrent d'une trajectoire requête. Les enquêteurs ont bien souvent des dizaines de vidéos à traiter à la fois. Parmi ces vidéos, certaines peuvent être filmées depuis des caméras proches et présenter du recouvrement. À partir d'une vidéo de départ, ils peuvent vouloir visualiser plus en détails une trajectoire suivie par un véhicule ou une personne, en naviguant de vidéo en vidéo à partir de requêtes trajectographiques.

Nous proposons ainsi de reformuler la trajectoire tracée par l'utilisateur dans la vidéo de départ dans toutes les autres caméras qui présentent du recouvrement. Il est assez improbable qu'une vidéo ne présentant aucun recouvrement avec la vidéo de départ parvienne à proposer une reformulation de trajectoire avec un score de reformulation R élevé : les rares corrélations apprises entre deux vues indépendantes peuvent être dues à des coïncidences de présence d'objets de même catégorie dans deux vues, et leurs valeurs sont généralement très faibles. Ainsi, les vues où la trajectoire requête donne un score de reformulation R faible peuvent être filtrées par seuillage. En pratique, nous considérons que l'on a préalablement déterminé quelles vidéos présentent du recouvrement, par exemple en utilisant notre approche présentée au chapitre 4.

Les trajectoires reformulées sont classées selon un critère. Dans notre cas, nous cherchons à offrir aux enquêteurs le meilleur point de vue possible pour voir la trajectoire avec un

maximum de détails et donc un maximum d'intérêt. Nous choisissons donc un critère de maximisation de la longueur de la trajectoire reformulée car nous faisons l'hypothèse que plus une trajectoire apparaît longue, plus elle est détaillée. Nous cherchons donc à maximiser ce que nous appelons le **score de visibilité**, défini comme le produit de leur **score de reformulation** R et de leur **longueur**. Dans la section suivante, nous présentons les expérimentations que nous avons menées pour évaluer la pertinence des vues proposées en fonction de plusieurs requêtes de trajectoire.

5.3.6 Évaluation des cartes de correspondance entre cellules

Nous souhaitons évaluer la qualité de nos cartes de correspondance, de nos reformulations de trajectoire, ainsi que la pertinence de notre classement des vues qui offrent une meilleure visualisation. Nous avons donc besoin d'un jeu de données contenant une collection de vidéos synchronisées avec du recouvrement dans leurs champs de vue. Il est préférable que les vidéos durent assez longtemps pour que les cartes de correspondance puissent être apprises : plus la durée commune des vidéos est importante, plus le risque de corréler les actions de régions qui ne se correspondent pas est faible.

Nous avons donc utilisé notre jeu de données ToCaDa [Malon 18] car il contient 19 vues qui présentent du recouvrement et plusieurs catégories (essentiellement piéton, moto et voiture) y sont présentes. Nous avons utilisé les annotations des catégories et des boîtes englobantes fournies dans le jeu de données pour calculer les cartes de correspondance des cellules. Les autres jeux de données multivues décrits au chapitre 3 présentent trop peu de points de vue différents pour qu'il soit pertinent de chercher à classer les meilleures vues depuis une vue courante.

Nous avons d'abord évalué la qualité des cartes de correspondance entre les 19 vidéos avec recouvrement. Nous estimons qu'une carte de correspondance est de bonne qualité si, lorsqu'à un instant donné, un même objet apparaît simultanément dans deux vidéos, les cartes de correspondance des cellules couvertes par la boîte englobante de l'objet dans la première vidéo offrent un score de correspondance élevé au niveau des cellules couvrant la boîte englobante de la deuxième vidéo. Plus formellement, nous définissons le **taux de correspondance** Γ entre une paire de boîtes englobantes B_1 et B_2 correspondant à un même objet vu simultanément dans V_1 et V_2 par :

$$\Gamma(B_1, B_2) = \sum_{(c_{\mathbf{i}}^{V_1}, c_{\mathbf{i}'}^{V_2})} \frac{|c_{\mathbf{i}}^{V_1} \cap B_1|}{|B_1|} \frac{|c_{\mathbf{i}'}^{V_2} \cap B_2|}{|B_2|} \frac{\mathcal{C}_\omega(c_{\mathbf{i}}^{V_1}, c_{\mathbf{i}'}^{V_2})}{||\mathcal{C}_\omega(c_{\mathbf{i}}^{V_1}, V_2)||}. \tag{5.4}$$

Le premier terme du produit pondère chaque cellule de la première vue par le pourcentage de la boîte englobante qu'elle couvre. Le second terme pondère de la même façon dans la deuxième vue. Le troisième terme pondère finalement par la corrélation entre paires de cellules issues des deux boîtes englobantes. Le taux de correspondance global est obtenu en

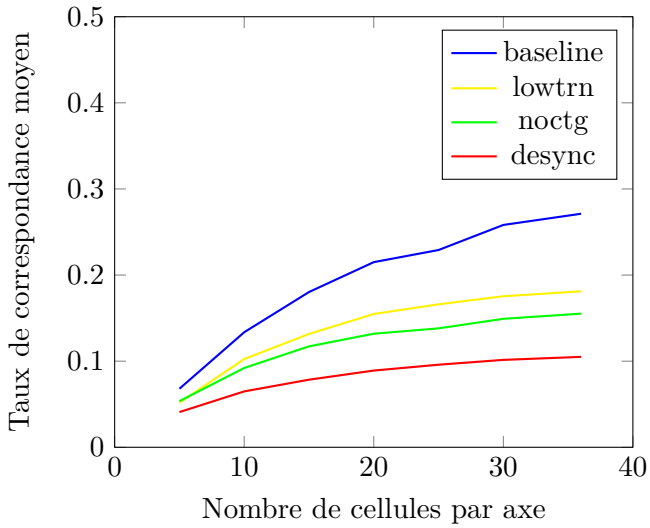


FIGURE 5.16 – Taux de correspondance moyen sur le jeu de données ToCaDa [Malon 18] pour différents nombres de cellules et différentes configurations : sans dégradation (baseline), sans distinction des catégories (noctg), avec une désynchronisation d'une seconde entre les vidéos (desync) et en apprenant les cartes de correspondance sur seulement la moitié du jeu de données (lowtrn).

moyennant l'expression (5.4) sur l'ensemble des boîtes englobantes simultanées d'un objet, puis en moyennant sur tous les objets communs à la paire de vidéos, puis en moyennant une dernière fois sur toutes les paires possibles de vidéos.

La figure 5.16 présente les taux de correspondance pour différentes configurations et différents nombres de cellules. Nous avons évalué le comportement de la méthode :

— lorsqu'il n'y a pas de dinstinction entre les catégories : chaque cellule dispose alors d'une seule fonction d'activité mesurant indifféremment la présence d'objets quelle que soit leur catégorie ;

— en créant un décalage d'une seconde entre les paires de vidéos ;

— en n'apprenant les cartes de correspondance que sur la moitié de la durée des vidéos.

Les résultats ont révélé que l'étape de calcul des cartes de correspondance est très sensible à ces perturbations, en particulier à la désynchronisation. On remarque également que la distinction des catégories permet d'améliorer grandement le taux de correspondance moyen. On ne s'attend bien entendu pas à avoir des taux de correspondance élevés dans la mesure où la carte de correspondance d'une cellule dans une vidéo couvre généralement un grand nombre de cellules dans l'autre vidéo. En revanche, nous espérons que ces taux de correspondance soient suffisants pour nous permettre d'effectuer correctement l'étape suivante : la reformulation.

5.3.7 Évaluation des reformulations de trajectoire

Pour mesurer la qualité de nos reformulations de trajectoire, nous avons tracé 10 trajectoires requêtes au niveau du sol dans différentes caméras et leur avons appliqué notre méthode de reformulation pour obtenir les trajectoires reformulées dans les autres caméras. Quelques unes de ces trajectoires requêtes sont montrées sur la figure 5.17.

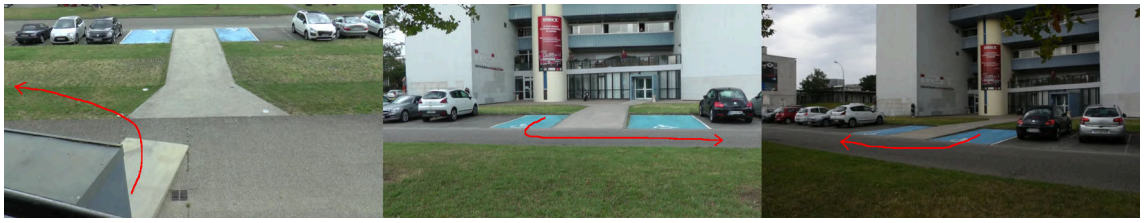


FIGURE 5.17 – Exemples de trajectoires requêtes.

Pour effectuer la comparaison par rapport à une trajectoire de référence, nous avons calculé l'homographie liée au plan du sol entre chaque paire de caméras à partir des coins des places de parking bleues. Nous avons échantillonné chaque trajectoire requête en 100 points espacés régulièrement, puis nous avons calculé les trajectoires reformulées théoriques dans chaque vue pour chacune des trajectoires requêtes en appliquant les homographies aux points échantillonnés. Nous avons de même échantillonné les trajectoires reformulées par notre approche en 100 points espacés régulièrement. Les points échantillonnés forment ce que l'on appellera une **séquence** dans laquelle ils apparaissent dans l'ordre. Pour mesurer la dissemblance entre les 100 points qui forment une trajectoire reformulée par notre approche et les 100 points qui forment une trajectoire reformulée de référence, nous avons utilisé l'algorithme de déformation temporelle dynamique ou *Dynamic Time Warping* (DTW) [Zhang 06]. DTW apparie les points de deux séquences selon les quatre règles suivantes :

1. chaque point d'une séquence doit être apparié à au moins un point de la deuxième séquence ;
2. le premier point de la première séquence doit être apparié au premier point de la deuxième séquence (et éventuellement à d'autres) ;
3. le dernier point de la première séquence doit être apparié au dernier point de la deuxième séquence (et éventuellement à d'autres) ;
4. l'appariement entre indices des points des deux séquences doit être croissant, c'est-à-dire que pour deux points de la première séquence d'indices $j > i$ et deux points dans la deuxième séquences d'indices $l > k$, on ne peut pas avoir de correspondances à la fois entre j et l et entre i et k.

La figure 5.18 illustre l'intérêt du choix de cette mesure. La quatrième règle se traduit par l'impossibilité d'avoir des croisements dans les appariements.

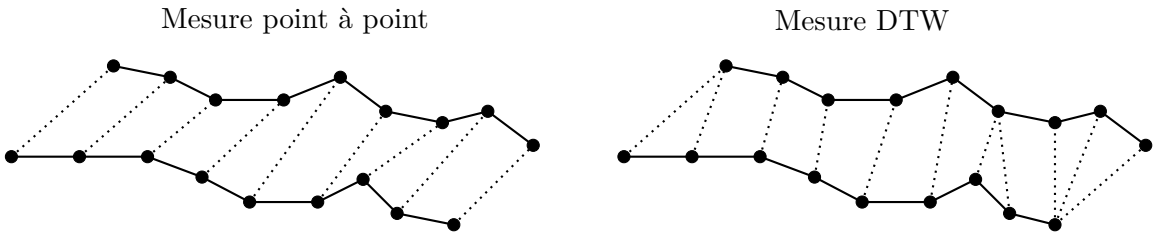


FIGURE 5.18 – Intérêt de la mesure DTW entre deux trajectoires – À partir de deux trajectoires échantillonnées par 9 points chacune, la mesure des distances point à point, présentée à gauche, est très sensible au décalage entre les trajectoires. La mesure DTW, à droite, est robuste à ce décalage : les distances sont calculées entre points les plus proches parmi les points d'indices supérieurs ou égaux à l'indice du dernier point apparié de chaque séquence.

La figure 5.19 présente la distance DTW moyenne, en pixels, sur des vidéos de taille 960×540 pour différents nombres de cellules. La trajectoire reformulée est plus fiable à mesure que le nombre de cellules augmente. En effet, comme nous obtenons la trajectoire reformulée en liant les centres des cellules, celle-ci peut s'approcher davantage de la trajectoire obtenue par homographie lorsqu'il y a davantage de cellules. Ces résultats montrent que notre méthode parvient à correctement reformuler les trajectoires.

Concernant le classement des vidéos, la figure 5.20 présente sur chaque ligne :

— à gauche, une trajectoire requête tracée manuellement dans une vue ;

— au milieu, les trois vidéos correspondant aux 3 meilleurs scores de visibilité, avec la trajectoire reformulée correspondante ;

— à droite, une vue classée parmi les dernières ; ces vues sont généralement des vues dans lesquelles on n'arrive pas à reformuler la totalité de la trajectoire requête, ou une vue où cette trajectoire reformulée occupe peu d'espace et ne permet pas de voir plus de détails.

Parmi les 5 meilleures vues proposées pour chacune des 10 trajectoires requêtes, nous avons voulu évalué la proportion d'entre elles qui offrent effectivement une meilleure ou aussi bonne visibilité de la trajectoire requête. Nous avons considéré qu'une vue offrait une meilleure visibilité si les deux conditions suivantes sont vérifiées :

— la trajectoire reformulée par le biais de notre approche est effectivement plus longue que la trajectoire requête ;

— la mesure DTW entre la trajectoire reformulée par notre approche et la trajectoire reformulée de référence est inférieure à un certain seuil.

Afin d'obtenir les trajectoires reformulées théoriques, nous avons calculé les homographies entre les différentes vues en recouvrement et appliqué ces homographies aux séquences de points des trajectoires requêtes. Le seuil de mesure DTW a été fixé à 10% de la longueur

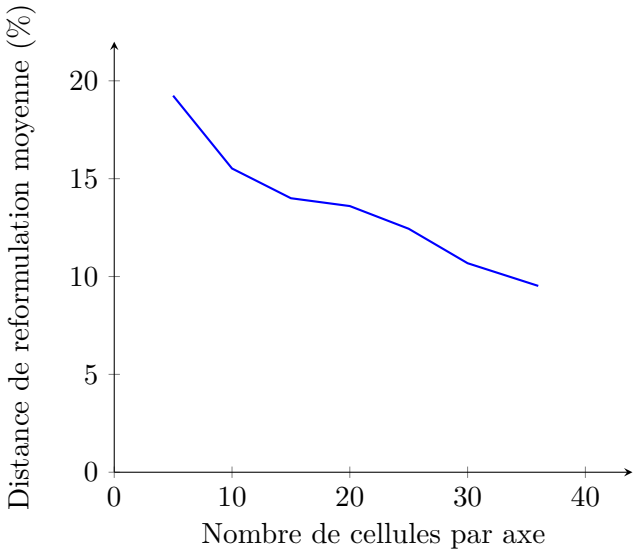


FIGURE 5.19 – Distance de reformulation moyenne DTW pour différents nombres de cellules – Les distances sont données comme un pourcentage de la longueur de la diagonale de l'image.

de la diagonale de l'image de sorte à être proportionnel à la taille de l'image et donc plus indulgent vis-à-vis d'images de grandes tailles. Nous avons trouvé que, parmi les 5 meilleures vues obtenues sur les 10 trajectoires requêtes, 72% d'entre elles donnent effectivement une meilleure ou aussi bonne visibilité de la trajectoire requête.

5.4 Conclusion

Dans ce chapitre, nous avons présenté la problématique de la visualisation des données, notamment dans le cas des images et des vidéos. Les méthodes existantes sont très variées selon que l'on cherche à visualiser un grand ensemble de données ou le contenu d'une seule vidéo. La problématique de la visualisation est très liée à celle de la navigation et des interactions. Comme on ne peut généralement pas afficher toutes les données à l'écran, il est nécessaire que l'utilisateur puisse interagir, soit par des actions simples à base de clics comme des translations, rotations ou zooms, soit par une requête plus complexe telle qu'une recherche par mot-clé, une esquisse du contenu recherché ou des contraintes en termes de position ou de trajectoire des objets. Dans un contexte d'enquête, nous avons proposé un type d'interaction original consistant à tracer une trajectoire dans une vidéo courante pour être redirigé vers une vidéo qui offre un point de vue depuis lequel cette trajectoire apparaît plus longue et donc plus détaillée. À partir d'une collection de vidéos et sans se servir des paramètres de calibrage des caméras, la méthode proposée permet de classer avec succès un sous-ensemble des vidéos qui présentent du recouvrement avec une

vidéo dans laquelle une requête trajectographique est tracée en fonction de la visibilité qu'elles offrent de la trajectoire reformulée. Les vidéos qui ne sont pas liées à la scène sont également correctement filtrées. Il n'est fait usage d'aucune méthode de réidentification, la simple présence simultanée d'objets de même catégorie dans différentes caméras permet d'apprendre les correspondances spatiales. Ces travaux ont été publiés à la conférence ICIP 2019 [Malon 19b].

Ce chapitre termine la présentation de nos contributions. Nous allons conclure et présenter nos perspectives dans le chapitre suivant.

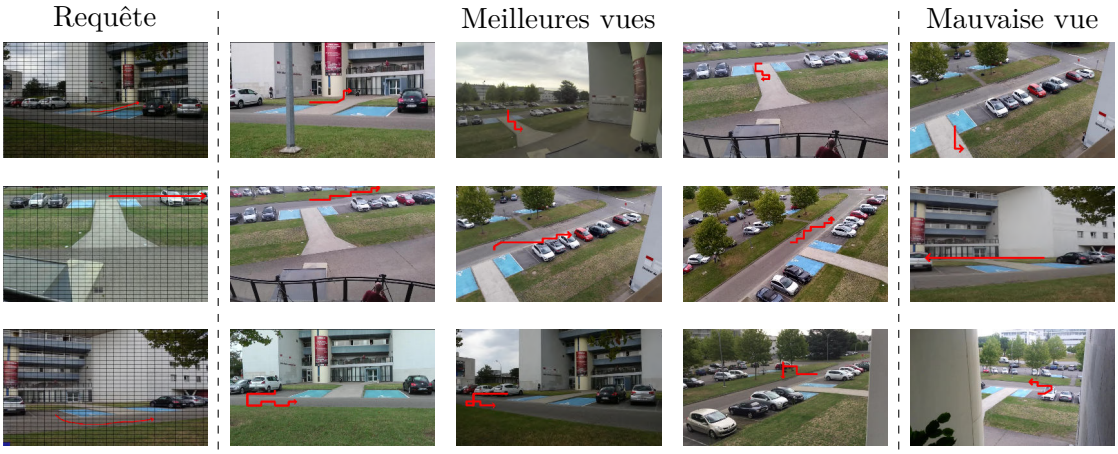


FIGURE 5.20 – Classement des meilleures vues – Colonne 1 : trois trajectoires requêtes sont tracées, en rouge, respectivement pour les catégories « piéton », « moto » et « voiture ». Colonnes 2 à 4 : les 3 vues qui offrent les meilleurs scores de visibilité sont affichées par ordre décroissant. Colonne 5 : une vue avec un faible score de visibilité.

Chapitre 6

Conclusion et perspectives

Contenu

6.1 Contributions à l'analyse de vidéos multiples

Nous l'avons mentionné dès le début de ce mémoire, cette thèse a été financée grâce au projet européen VICTORIA dont l'objectif est de proposer des outils d'aide à l'analyse de vidéos multiples dans un contexte d'enquête. Nous n'avons pas détaillé les travaux directement liés aux engagements de livrables liés à ce projet car ils ne présentent pas de dimension recherche. Ils supposent un contexte très contrôlé dans lequel les paramètres des différentes caméras sont connus. De plus, les tâches assignées à chaque équipe étaient précises et ciblées. Dans notre cas, certains traitements, tels que la détection, le suivi ou la réidentification, étaient réalisés par une autre équipe et leurs résultats étaient fournis en entrée de nos algorithmes. Toutefois, nous souhaitons présenter brièvement les éléments étudiés et produits dans ce cadre dans la section 6.1.1, avant de reprendre toutes les contributions. Nous distinguons les contributions collectives à plusieurs équipes liées au projet, listées dans la section 6.1.2, des contributions présentées dans ce mémoire et rappelées dans la section 6.1.3.

6.1.1 Implication dans le projet VICTORIA

Dans le cadre du projet VICTORIA, nous avons participé au lot qui concerne l'analyse vidéo et la reconstruction 4D de scène. Une équipe partenaire allemande du laboratoire de l'institut Fraunhofer-Gesellschaft était en charge de la reconstruction 3D des parties statiques et du développement d'une interface permettant de lancer la reconstruction et d'en visualiser le rendu. Notre rôle était d'enrichir cette reconstruction avec les parties dynamiques des objets rigides, essentiellement des véhicules, et des piétons, représentés par des squelettes 3D se déplaçant au fil du temps. Ces squelettes 3D sont composés d'un ensemble de points, les jonctions, correspondant à différentes articultations et parties du corps humain, notamment les poignets, les coudes les épaules et la tête. La tâche qui nous était assignée consistait à estimer les points de jonction en 2D en utilisant l'algorithme `OpenPose`[1] [Cao 19] sur des imagettes de piétons issues de plusieurs caméras, puis à trianguler les jonctions 2D correspondant à un même piéton dans différentes caméras — l'étape de réidentification étant réalisée par une équipe du lot 5 — en points de jonction 3D. Les imagettes étaient issues des suivis des différents piétons, calculés par cette même équipe du lot 5 et stockés dans des fichiers contenant une suite de boîtes englobantes correspondant aux positions successives d'un même piéton au cours du temps. Concernant les objets rigides de type véhicule, la triangulation se faisait de façon plus grossière en considérant l'objet entier et en calculant des intersections entre cônes projetés au travers

1. https://github.com/CMU-Perceptual-Computing-Lab/openpose

des ellipses d'aires maximales incluses dans les boîtes englobantes dans les différentes caméras. À partir de ces intersections, nous avons obtenu un nuage de points qui a permis d'estimer une forme grossière de l'objet, dans notre cas un ellipsoïde. La réidentification entre piétons ou objets rigides était fournie : les fichiers de suivi d'un même objet dans des caméras différentes portaient le même nom. Enfin, nous disposions des paramètres de calibrage des différentes caméras et des transformations homographiques du plan du sol entre les différentes vidéos.

En pratique, les suivis des piétons ont été difficiles à exploiter car ils présentaient de nombreuses incohérences : lorsque deux piétons se croisaient, il arrivait régulièrement que les suivis des deux personnes s'échangent, que l'un des deux disparaisse, ou encore qu'un nouveau fichier de suivi soit créé comme si une nouvelle personne était apparue. De la même façon, les réidentifications, basées sur l'hypothèse qu'un piéton correspond à un et un seul fichier de suivi par caméra, se sont révélées en partie erronées et n'ont pas été utilisées. Pour parvenir à faire correspondre les piétons de différentes caméras, nous avons donc utilisé, à chaque instant t des vidéos synchronisées, chaque boîte englobante de chaque suivi individuellement. Pour chaque boîte englobante, nous avons considéré son point au niveau du sol, comme illustré lors du chapitre 2, puis nous lui avons appliqué les transformations homographiques entre vidéos pour obtenir sa position théorique dans les autres vidéos. Nous avons ensuite apparié les points transformés théoriques avec les points au niveau du sol des boîtes englobantes. Cette réidentification purement géométrique s'est avérée efficace : le fait de considérer les boîtes englobantes de façon individuelle et non plus sous forme de suivis limite les erreurs de correspondance, au prix d'appliquer la transformation homographique à chaque boîte englobante de chaque image. Enfin, nous corrigeons les jonctions 3D obtenues en considérant différentes longueurs entre jonctions : à titre d'exemple, la longueur entre l'épaule et le coude droits doit être similaire à celle entre l'épaule et le coude gauches.

Le fait de ne pas exploiter l'apparence de l'objet ou de la personne dans l'étape de réidentification présente un avantage majeur vis-à-vis des aspects juridiques. Au cours du projet, des problématiques sur le droit à l'image ont été soulevées par un partenaire du *KU Leuven Centre for IT and IP Law*, notamment sur le stockage des vidéos dans lesquelles apparaissent des visages ou des plaques d'immatriculation et sur l'accès à ces vidéos. Le constat était que les opérateurs travaillant sur ces vidéos n'ont pas tous besoin du même niveau d'information. Les solutions proposées incluaient la possibilité de mettre en place différents niveaux d'accès n'autorisant un opérateur à accéder qu'à l'information dont il a besoin ou à laquelle il a le droit d'accéder, notamment en floutant ou en masquant les visages et les plaques d'immatriculation le cas échant. Pour cette raison, notre méthode de réidentification, bien qu'assez simpliste, ne subit pas l'impact de ces restrictions.

En dehors de ces travaux de développement, nous avons rédigé trois documents livrables. Le premier porte sur les spécifications techniques, notamment sur les formats et la struc-

ture des fichiers attendus en entrée, ainsi que sur l'arborescence des fichiers. Le deuxième détaille le calcul des points d'intersection entre cônes pour les objets rigides et l'estimation d'un ellipsoide à partir de ces points. Il présente également l'algorithme `OpenPose` pour le calcul des points de jonction 2D sur des imagettes de piétons et la méthode de triangulation 3D de ces points. Enfin, le troisième document livrable a été coécrit avec l'équipe allemande en charge de la reconstruction des parties statiques de la scène et détaille le fonctionnement de leur interface et l'intégration des différents modules. Pour des raisons de confidentialité, nous ne pouvons pas fournir d'illustration de cette interface ni d'image des résultats de nos travaux tirée de ces livrables.

Enfin, le projet VICTORIA a été ponctué de réunions et d'évènements à intervalles plus ou moins réguliers :

— des réunions téléphoniques hebdomadaires avec les partenaires de lot lors de certaines périodes de développement ;

— des rapports d'avancement à fournir tous les trois mois ;

— des réunions annuelles sur plusieurs journées en présence de tous les partenaires dans les locaux de l'un d'entre eux (en mai 2018, nous avons accueilli la réunion annuelle à l'IRIT) ;

— quatre *hackathons* à Bochum, en Allemagne, au cours desquels l'ensemble des partenaires techniques se sont réunis pour intégrer les différents modules dans une plateforme commune et en tester le bon fonctionnement ;

— deux réunions d'évaluation par la commission européenne.

Bien que le travail de reconstruction des parties dynamiques de la scène, effectué pour VICTORIA, n'ait pas contribué directement à cette thèse, le projet aura été une expérience internationale enrichissante. Il a également donné un contexte de travail : celui des enquêtes policières. Ce contexte, riche et intéressant, nous a conduits à travailler de concert avec deux autres équipes de l'IRIT. Cette collaboration a donné lieu à des publications collectives.

6.1.2 Contributions collectives à la thématique des enquêtes

En dehors des contributions détaillées dans ce mémoire de thèse et qui seront rappelées dans la section suivante, trois autres publications ont été menées avec les équipes SAMOVA et SIG de l'IRIT impliquées elles aussi sur des thématiques d'enquêtes. Ces équipes travaillent respectivement sur l'audio et sur les métadonnées de vidéos. Nous avons réalisé et annoté un jeu de données multicaméras que nous avons appelé ToCaDa (pour *Toulouse Campus Dataset*). Il a été constitué à partir de 25 caméras dont 19 présentent du recouvrement dans leurs champs de vue. Les autres ont des points de vue disjoints. Nous l'avons beaucoup employé dans nos expériences, de par le grand nombre de caméras en recouvre-

ment qu'il contient. Ce jeu de données tient compte des problématiques des différentes équipes impliquées : la partie sonore y est annotée au même titre que la partie visuelle, et certaines métadonnées complémentaires, telles que les coordonnées GPS de certaines caméras, ont été relevées. Il a été publié à la conférence MMSys 2018 [Malon 18] et mis en avant sur le site de ACM SIGMM [2] .

L'étape d'annotation de ce jeu de données a soulevé des questionnements sur la façon de lier les annotations audio et les annotations vidéo. Lorsqu'un véhicule est présent à l'écran et que l'on entend un son de klaxon, on peut vouloir lier l'objet visuel, par exemple une boîte englobante autour du véhicule, à l'objet sonore entendu, le son de klaxon. À partir de cette affirmation de base, nous avons considéré un ensemble d'autres configurations. Si l'on ne dispose d'aucune autre information sur la scène, rien ne nous assure que c'est bien le véhicule observé qui a produit le son entendu, il peut s'agir d'un autre véhicule qui n'apparaît pas dans le champ de vue. Si l'on imagine deux véhicules se percutant, comment lier l'objet sonore correspondant au son de l'impact et les deux objets visuels correspondant à chacun des véhicules ? Nous avons proposé une réponse dans l'article [Guyot 19] publié à la conférence MMM 2019 en définissant la notion d'objet audiovisuel et en donnant une catégorisation des différents types de liens entre objets audiovisuels.

Enfin, une troisième contribution commune porte sur la réidentification de véhicules en utilisant des représentations latentes, c'est-à-dire en utilisant l'avant-dernière couche d'un réseau de neurones convolutif en tant que descripteur. Dans cette contribution, nous nous plaçons dans un contexte où l'on dispose de plusieurs imagettes par objet. Nous envisageons ainsi différents scénarios dans lesquels les réponses sont composées de plusieurs imagettes, à savoir celui que nous qualifions de *image-to-track*, où la requête est formée d'une seule imagette et celui que nous qualifions de *track-to-track* où la requête est formée de plusieurs imagettes. Nous avons étudié l'impact du choix de la mesure en comparant la distance euclidenne et la distance cosinus, et celui du choix du réseau utilisé sur les performances en termes de réidentification. De manière générale, nos résultats mettent en évidence que l'on obtient de meilleures performances lorsque la requête est composée de plusieurs imagettes. D'autre part, les meilleures performances ont été obtenues en utilisant la distance cosinus combinée au réseau DenseNet201 [Huang 17]. Ce travail a été publié dans la revue IET en 2020 [Roman-Jimenez 20].

6.1.3 Contributions personnelles à l'analyse de vidéos multiples

Au cours de cette thèse, nous avons traité le problème de l'aide à la compréhension d'une collection de vidéos. Cette aide s'est traduite sous plusieurs formes :

— au travers d'une recherche automatique de liens entre vidéos ;

2. http://records.sigmm.org/2020/03/05/dataset-column-tocada-dataset-with-multi-viewpoint-synchronized-videos/

— au travers d'une collaboration entre l'humain et la machine pour affiner les liens trouvés ;

— par un mécanisme de navigation entre vidéos en formulant des requêtes de trajectoires.

Chacun de ces aspects a fait l'objet d'une publication scientifique. Nous avons présenté les contributions dans un ordre diffèrent de l'ordre de publication car nous trouvons qu'il donne un meilleur fil conducteur. Notre première contribution s'est appliquée à déterminer les liens entre vidéos en extrayant leurs contenus sous une forme synthétique que nous avons nommée l'histoire de la vidéo, puis en comparant ces histoires. Cette approche se distingue des approches déjà existantes par le fait qu'elle ne s'appuie pas sur des réidentifications fines entre les objets qui apparaissent dans les vidéos, mais sur la cohérence globale des apparences et des catégories de ces objets à différentes échelles spatiales et temporelles. Nous avons obtenu de bons résultats en appliquant cette approche à une collection d'une soixantaine de vidéos en retrouvant la majorité des liens de recouvrement entre caméras. Cette première contribution a été acceptée pour publication à la conférence ICPR 2020 [Malon 20b]. Elle a aussi été publiée en français à la conférence RFIAP de cette même année [Malon 20a].

Les histoires de vidéos dont la distance se situe au voisinage des seuils d'apparence ou de proportion présentent le risque qu'un lien soit trouvé à tort ou à l'inverse qu'un lien erroné soit trouvé. Pour gérer cet aspect, l'apprentissage actif mis en place dans notre deuxième contribution introduit la possibilité de collaboration entre l'humain et la machine : par son expertise, nous avons montré que l'humain parvient à apprendre à la machine à trouver les liens entre vidéos à partir de quelques exemples et sollicitations. En comparaison avec des méthodes entièrement automatiques, le fait d'inclure l'humain dans le processus améliore les résultats. Nous prévoyons de soumettre cette deuxième contribution dans la revue CVM fin 2020.

En termes de navigation entre vidéos, nous avons proposé un mécanisme de requête trajectographique permettant de passer d'une vidéo courante à une vidéo proposant une visualisation différente que nous supposons plus intéressante pour un utilisateur. Ce dernier s'appuie sur une reformulation de la trajectoire tracée en son équivalent dans les autres vues obtenue grâce aux correspondances entre cellules aux profils d'activité similaires. Nous avons évalué les aspects de reformulation de trajectoire et de pertinence des réponses aux requêtes : de manière générale, les trajectoires sont bien reformulées et les vidéos atteintes après une requête offrent un meilleur point de vue pour visualiser la trajectoire. Cette troisième contribution a été publiée à la conférence ICIP 2019 [Malon 19b]. Une version traduite en français a également été proposée aux journées ORASIS la même année [Malon 19a].

6.2 Perspectives

6.2.1 Amélioration des méthodes proposées

En termes d'améliorations directes des méthodes proposées, nous souhaitons tester d'autres caractéristiques pour décrire les histoires des vidéos. Le son semble être un indicateur pertinent pour la recherche de liens entre caméras : si les caméras sont proches, il est probable qu'un son enregistré par l'une soit aussi enregistré par l'autre. Les détections sonores ne sont cependant pas assignables spatialement à des cellules ; notre approche multirésolution serait alors à adapter. Une autre piste que nous voulons développer s'appuie sur les correspondances trouvées entre cellules aux histoires similaires. À partir des appariements de cellules les plus forts, il semble possible d'estimer une transformation homographique entre plans du sol de différentes caméras puis de trouver d'autres correspondances entre cellules grâce à cette homographie, notamment entre cellules dans lesquelles aucun objet n'a été détecté. Enfin, nous voulons ajouter un système de pondération à notre mesure de distance entre histoires en donnant plus de poids aux catégories d'objets les moins communes, ou en considérant des successions d'objets.

Concernant la recherche de liens par apprentissage actif, nous souhaitons améliorer notre prototype d'interface et le rendre plus simple et intuitif à utiliser. Ensuite, nous souhaitons entreprendre une étude utilisateur pour mesurer la qualité d'expérience et la qualité de service des utilisateurs de cette interface. Cette expérience sera aussi l'occasion d'identifier les vidéos difficiles à classer par un oracle humain. En effet, dans nos expériences, nous avons supposé que l'oracle ne peut pas se tromper lorsqu'il est sollicité. En pratique, l'oracle a une certaine probabilité de faire une erreur de classification. Cette probabilité dépend de l'expertise de l'oracle, de la vidéo à classer et des vidéos qui ont déjà été classées préalablement et par rapport auxquelles l'oracle peut se repérer.

Enfin, concernant la navigation entre vidéos via des requêtes, nous souhaitons proposer d'autres types de requêtes que les tracés de trajectoires. L'une des requêtes que nous voudrions proposer est la possibilité, depuis une vidéo dans laquelle des caméras sont visibles, de sélectionner l'une des caméras, et d'être redirigé vers la vidéo de cette caméra, ou si elle n'est pas disponible, la vidéo filmée depuis le point de vue le plus similaire. Selon l'un des enquêteurs avec qui nous avons pu échanger au cours du projet, ce type d'interaction serait particulièrement utile, notamment lors d'enquêtes en lien avec des incidents survenus pendant une manifestation. Les enquêteurs peuvent avoir recours à des vidéos filmées par des témoins pour retrouver le lanceur d'un projectile. S'il est difficile à identifier dans une vidéo, mais qu'ils voient dans cette même vidéo une autre personne en train de filmer depuis un point de vue plus favorable, ils savent alors qu'il existe une vidéo dans laquelle le lanceur sera plus facilement identifiable.

6.2.2 Continuité des méthodes proposées

L'originalité des travaux que nous avons menés repose sur l'étude de la corrélation des activités entre vidéos, sans utiliser d'outils de suivi ou de réidentification. Nous avons fait ce choix initial car nous avons estimé que nous n'avions pas les conditions idéales pour exploiter ces outils. Les objets présents dans les vidéos étaient de trop basses résolutions et surtout, les positions des caméras et leurs points de vue étaient trop différents. La détection d'objets est donc délicate et il est d'autant plus délicat d'en extraire un suivi fiable. Les résultats que nous avons présentés avec cette approche sans suivi et sans réidentifcation sont encourageants mais nous souhaitons les enrichir. À l'issue de cette thèse, les perspectives que nous envisageons pour enrichir nos travaux et permettre une ouverture sur d'autres types d'approches sont présentées dans les sections 6.2.3 à 6.2.7. Elles s'appuient notamment sur la figure 6.1.

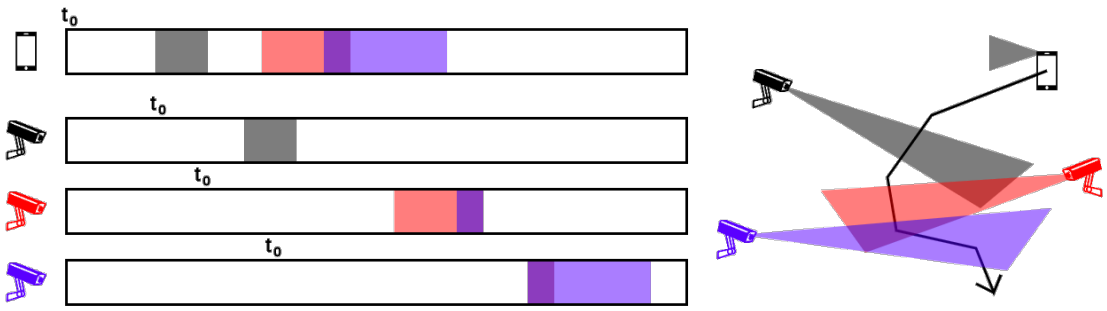


FIGURE 6.1 – Analyse de réseaux de caméras désynchronisées avec caméras mobiles – Nous illustrons les types de configuration qui n'ont pas été abordés dans nos travaux et que nous aimerions aborder. Cet exemple s'appuie sur un cas avec 3 caméras fixes et 1 caméra mobile, liée à un smartphone. Nous supposons ainsi que le champ de vue de la caméra du smartphone va croiser celui des autres caméras sur le chemin qu'il va emprunter, matérialisé par la flèche noire. Cet exemple illustre également le cas des caméras qui n'ont pas des champs de vue qui se recoupent mais où il y a un lien car il est fort probable que les objets vus par la caméra noire soient vus par la caméra bleue après un certain délai. Enfin, les frises de temps indiquent les moments de chaque vidéo où le champ de vue de la caméra du smartphone croise le champ de vue des autres caméras et illustrent un cas où les capteurs ne sont pas synchronisés sur une même horloge. Une date commune t_0 peut alors correspondre à différents instants dans chacune d'entre elles.

6.2.3 Gestion des vidéos non synchronisées

Dans nos travaux, nous avons supposé une synchronisation temporelle des vidéos selon une même horloge. En pratique, les vidéos qu'exploitent des enquêteurs, notamment celles provenant de témoins, ne sont pas synchronisées entre elles. De la même façon que pour la gestion des vidéos provenant de capteurs mobiles, nous devons envisager dans un premier temps de trier les vidéos en groupes de vidéos synchronisées ou faciles à synchroniser

(ce sont celles pour lesquelles nous avons l'information d'horodatage fiable) entre elles pour ensuite appliquer une stratégie de traitement de ces désynchronisations. Pour les vidéos où il est nécessaire de faire un travail de synchronisation avec les autres, nous pouvons imaginer utiliser la méthode que nous avons déjà mise en œuvre de comparaison des histoires, mais, cette fois, en effectuant une recherche sur des portions de la vidéo et, d'autre part, en essayant des voisinage temporels beaucoup plus grands. Cela signifie que nous devons relever les défis suivants : être capable de déterminer un intervalle de temps suffisamment pertinent pour établir une comparaison et être capable de déterminer une stratégie de recherche rapide, car, si nous appliquons telle quelle la méthode actuelle, les temps de calcul seront très élevés.

6.2.4 Analyse des vidéos ayant un lien sans recouvrement

Dans cette thèse, nous avons principalement traité le cas des vidéos liées par des champs de vue en recouvrement. Il est nécessaire d'ajouter les liens entre vidéos sans recouvrement mais avec un lien de dépendance, comme c'est le cas entre les caméras noire et bleue de la figure 6.1. Pour effectuer cette tâche, nous pouvons à nouveau travailler sur la notion d'histoire des vidéos. En effet, jusqu'à présent, nous avons travaillé sur la concomitance des détections en cherchant une cohérence globale sur toute la durée de la vidéo. À présent, il faut explorer cette concomitance d'activité mais avec un intervalle de temps suffisamment significatif pour dire qu'il y a bien une succession et non une simultanéité des activités. Cette tâche est délicate car elle augmente considérablement les calculs à réaliser. Toutefois, il paraît évident que ce travail doit être fait après avoir établi les liens directs et donc, cela réduira les temps de calcul. Cette description nous permet de mettre en évidence que ce traitement pourrait être assez lié ou similaire à celui que nous envisageons pour la gestion des vidéos non synchronisées.

6.2.5 Prise en compte de la détection des évènements

Jusqu'à présent, nous avons étudié toutes les activités sans chercher à les qualifier. En traitement de la vidéo, il existe de nombreux travaux qui portent sur l'identification d'images clés, *key frames*, pour extraire les moments les plus significatifs dans une vidéo [Liu 03], ou encore la recherche des éléments clés d'une scène, les éléments qui contiennent l'information la plus pertinente [Bellver 16]. Notre approche pourrait être adaptée en donnant plus d'importance à ces évènements pour identifier les concomitances d'évènements plutôt que de toutes les activités. Cette prise en compte des évènements devrait également nous permettre de trouver plus rapidement les liens entre vidéos sans recouvrement.

6.2.6 Gestion des caméras mobiles

La gestion des caméras mobiles implique une philosophie de traitement assez différente de celle que nous avons mise en place. Intuitivement, nous pouvons supposer qu'une première tâche consistera à trier les vidéos selon qu'elles proviennent de caméras mobiles ou statiques. Cette tâche peut être réalisée en étudiant le mouvement global dans la vidéo. Le fait que la caméra soit mobile implique que le recouvrement entre vidéos ne sera pas constant et stable, comme pour le cas des caméras statiques. Cela implique également que le temps d'accumulation des évidences de recouvrement sera beaucoup plus court. Toutes ces difficultés indiquent qu'il sera nécessaire de donner plus d'importance à la reconnaissance et à la réidentification d'objets, d'une part, et qu'il faudra s'appuyer sur un apprentissage plus conséquent, d'autre part. Enfin, il s'agit d'un cas limite où l'utilisateur devra jouer une part plus importante en indiquant éventuellement un objet d'intérêt repéré dans une vidéo. Ces vidéos provenant de capteurs mobiles pourront donc être utilisées en complément des autres vidéos dans lesquelles un utilisateur indiquera un objet d'intérêt.

6.2.7 Ajout de la notion de profondeur

Les travaux présentés dans ce mémoire s'appuient sur un découpage spatial des vidéos en cellules régulières. La notion de profondeur n'intervient pas. Son utilisation devrait permettre d'améliorer les résultats. En effet, actuellement, il est possible d'estimer la profondeur, même dans des conditions non idéales, en utilisant des réseaux de neurones antagonistes génératifs, comme dans [Abdulwahab 20]. Le travail que nous aimerions aborder consiste à définir comment combiner cette information 3D approximative avec ce que nous avons déjà mis en place. Nous pouvons imaginer que cette information de profondeur donne une information sur les objets mais également sur les décors afin d'estimer la corrélation des personnes et des objets. Cette information pourrait participer au calcul de distance que nous utilisons. Nous pouvons d'ailleurs citer l'article [Ruiz-Hidalgo 12] qui propose de combiner couleur et profondeur de caméras multiples pour réaliser une segmentation de la scène. De manière générale, être en capacité d'analyser les différents éléments statiques et dynamiques de la scène, en s'appuyant sur une reconstruction partielle de la scène, devrait nous permettre d'améliorer l'approche proposée.

Bibliographie

[Abdulwahab 20] Saddam Abdulwahab, Hatem A Rashwan, Miguel Angel Garcia, Mohammed Jabreel, Sylvie Chambon et Domenec Puig. *Adversarial Learning for Depth and Viewpoint Estimation from a Single Image. IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[Arthur 06] David Arthur et Sergei Vassilvitskii. k-means++ : The advantages of careful seeding. Rapport Technique, Stanford, 2006.

[Barthel 15] Kai Uwe Barthel, Nico Hezel et Radek Mackowiak. *Graph-based browsing for large video collections.* Dans International Conference on Multimedia Modeling, 2015.

[Barthel 17] Kai Uwe Barthel et Nico Hezel. *Graph navigation for exploring very large image collections.* Dans International Conference on Computer Vision Theory and Applications, 2017.

[Baum 91] Eric B Baum. *Neural net algorithms that learn in polynomial time from examples and queries. IEEE Transactions on Neural Networks*, 1991.

[Bellver 16] Miriam Bellver, Xavier Giró, Ferran Marqués et Jordi Torres. *Hierarchical object detection with deep reinforcement learning. arXiv preprint arXiv:1611.03718*, 2016.

[Boser 92] Bernhard E Boser, Isabelle M Guyon et Vladimir N Vapnik. *A training algorithm for optimal margin classifiers.* Dans Workshop on Computational Learning Theory, 1992.

[Cao 19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei et Y. A. Sheikh. *OpenPose : Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[Carlier 17] Axel Carlier, Lilian Calvet, Pierre Gurdjos, Vincent Charvillat et Wei Tsang Ooi. *Querying multiple simultaneous video streams with 3D interest maps.* Dans Visual Content Indexing and Retrieval with Psycho-Visual Models. 2017.

[Caspi 06] Yaron Caspi, Anat Axelrod, Yasuyuki Matsushita et Alon Gamliel. *Dynamic stills and clip trailers. The Visual Computer*, 2006.

[Chambon 11] Sylvie Chambon et Alain Crouzil. *Similarity measures for image matching despite occlusions in stereo vision. Pattern Recognition*, 2011.

[Chavdarova 18] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool et François Fleuret. *WILDTRACK : A multi-camera HD dataset for dense unscripted*

pedestrian detection. Dans Conference on Computer Vision and Pattern Recognition, 2018.

[Chen 11a] Daozheng Chen, Mustafa Bilgic, Lise Getoor, David Jacobs, Lilyana Mihalkova et Tom Yeh. *Active inference for retrieval in camera networks.* Dans Workshop on Person-Oriented Vision, 2011.

[Chen 11b] Kuan-Wen Chen, Chih-Chuan Lai, Pei-Jyun Lee, Chu-Song Chen et Yi-Ping Hung. *Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. IEEE Transactions on Multimedia*, 2011.

[Cho 19] Yeong-Jun Cho, Su-A Kim, Jae-Han Park, Kyuewang Lee et Kuk-Jin Yoon. *Joint person re-identification and camera network topology inference in multiple cameras. Computer Vision and Image Understanding*, 2019.

[Comaniciu 02] Dorin Comaniciu et Peter Meer. *Mean shift : A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence*, 2002.

[Cortes 95] Corinna Cortes et Vladimir Vapnik. *Support-vector networks. Machine Learning*, 1995.

[Cover 67] Thomas Cover et Peter Hart. *Nearest neighbor pattern classification. Transactions on Information Theory*, 1967.

[Dalal 05] Navneet Dalal et Bill Triggs. *Histograms of oriented gradients for human detection.* Dans Conference on Computer Vision and Pattern Recognition, 2005.

[Deng 09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei. *Imagenet : A large-scale hierarchical image database.* Dans Conference on Computer Vision and Pattern Recognition, 2009.

[Detmold 07] Henry Detmold, Anton Van Den Hengel, Anthony Dick, Alex Cichowski, Rhys Hill, Ekim Kocadag, Katrina Falkner et David S Munro. *Topology estimation for thousand-camera surveillance networks.* Dans International Conference on Distributed Smart Cameras, 2007.

[Everingham 10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn et Andrew Zisserman. *The Pascal visual object classes (voc) challenge. International Journal of Computer Vision*, 2010.

[Fei-Fei 04] Li Fei-Fei, Rob Fergus et Pietro Perona. *Learning generative visual models from few training examples : An incremental bayesian approach tested on 101 object categories.* Dans Conference on Computer Vision and Pattern Recognition, 2004.

[Felzenszwalb 04] Pedro F Felzenszwalb et Daniel P Huttenlocher. *Efficient graph-based image segmentation. International Journal of Computer Vision*, 2004.

[Felzenszwalb 05] Pedro F Felzenszwalb et Daniel P Huttenlocher. *Pictorial structures for object recognition. International Journal of Computer Vision*, 2005.

[Felzenszwalb 08] Pedro Felzenszwalb, David McAllester et Deva Ramanan. *A discriminatively trained, multiscale, deformable part model.* Dans Conference on Computer Vision and Pattern Recognition, 2008.

[Felzenszwalb 09] Pedro F Felzenszwalb, Ross B Girshick, David McAllester et Deva Ramanan. *Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence*, 2009.

[Ferryman 09] James Ferryman et Ali Shahrokni. *Pets2009 : Dataset and challenge.* Dans International Workshop on Performance Evaluation of Tracking and Surveillance, 2009.

[Fischler 73] Martin A Fischler et Robert A Elschlager. *The representation and matching of pictorial structures. IEEE Transactions on Computers*, 1973.

[Fleuret 07] Francois Fleuret, Jerome Berclaz, Richard Lengagne et Pascal Fua. *Multicamera people tracking with a probabilistic occupancy map. Pattern Analysis and Machine Intelligence*, 2007.

[Gilbert 08] Andrew Gilbert et Richard Bowden. *Incremental, scalable tracking of objects inter camera. Computer Vision and Image Understanding*, 2008.

[Girshick 14] Ross Girshick, Jeff Donahue, Trevor Darrell et Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation.* Dans Conference on Computer Vision and Pattern Recognition, 2014.

[Girshick 15] Ross Girshick. *Fast R-CNN.* Dans International Conference on Computer Vision, 2015.

[Guyon 12] Charles Guyon, Thierry Bouwmans, El-hadi Zahzah et autres. *Robust principal component analysis for background subtraction : Systematic evaluation and comparative analysis. Principal Component Analysis*, 2012.

[Guyot 19] Patrice Guyot, Thierry Malon, Geoffrey Roman-Jimenez, Sylvie Chambon, Vincent Charvillat, Alain Crouzil, André Péninou, Julien Pinquier, Florence Sèdes et Christine Sénac. *Audiovisual annotation procedure for multi-view field recordings.* Dans International Conference on Multimedia Modeling, 2019.

[He 16] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. *Deep residual learning for image recognition.* Dans Conference on Computer Vision and Pattern Recognition, 2016.

[He 17] Kaiming He, Georgia Gkioxari, Piotr Dollár et Ross Girshick. *Mask R-CNN.* Dans International Conference on Computer Vision, 2017.

[Höferlin 09] Markus Höferlin, Benjamin Höferlin et Daniel Weiskopf. *Video visual analytics of tracked moving objects.* Dans Workshop on Behaviour Monitoring and Interpretation, 2009.

[Höferlin 11] Markus Höferlin, Benjamin Höferlin, Daniel Weiskopf et Gunther Heidemann. *Interactive schematic summaries for exploration of surveillance video.* Dans ACM International Conference on Multimedia Retrieval, 2011.

[Huang 17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten et Kilian Q Weinberger. *Densely connected convolutional networks.* Dans Conference on Computer Vision and Pattern Recognition, 2017.

[Jojic 09] Nebojsa Jojic, Alessandro Perina, Marco Cristani, Vittorio Murino et Brendan Frey. *Stel component analysis : Modeling spatial correlations in image class structure.* Dans Conference on Computer Vision and Pattern Recognition, 2009.

[Joshi 09] Ajay J Joshi, Fatih Porikli et Nikolaos Papanikolopoulos. *Multi-class active learning for image classification.* Dans Conference on Computer Vision and Pattern Recognition, 2009.

[Kang 06] Hong-Wen Kang, Yasuyuki Matsushita, Xiaoou Tang et Xue-Quan Chen. *Space-time video montage.* Dans Conference on Computer Vision and Pattern Recognition, 2006.

[Karlos 19] Stamatis Karlos, Vasileios G Kanas, Christos Aridas, Nikos Fazakis et Sotiris Kotsiantis. *Combining Active Learning with Self-train algorithm for classification of multimodal problems.* Dans International Conference on Information, Intelligence, Systems and Applications, 2019.

[Khan 03] Sohaib Khan et Mubarak Shah. *Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. Pattern Analysis and Machine Intelligence*, 2003.

[Khoshrou 14] Samaneh Khoshrou, Jaime S Cardoso et Luis F Teixeira. *Active learning from video streams in a multi-camera scenario.* Dans Conference on Computer Vision and Pattern Recognition, 2014.

[Koestinger 12] Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M Roth et Horst Bischof. *Large scale metric learning from equivalence constraints.* Dans Conference on Computer Vision and Pattern Recognition, 2012.

[Krizhevsky 12] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* Dans Advances in Neural Information Processing Systems, 2012.

[Land 71] Edwin H Land et John J McCann. *Lightness and retinex theory. Josa*, 1971.

[Lewis 94] David D Lewis et William A Gale. *A sequential algorithm for training text classifiers.* Dans Special Interest Group on Information Retrieval, 1994.

[Liao 15] Shengcai Liao, Yang Hu, Xiangyu Zhu et Stan Z Li. *Person re-identification by local maximal occurrence representation and metric learning.* Dans Conference on Computer Vision and Pattern Recognition, 2015.

[Lin 14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C Lawrence Zitnick. *Microsoft COCO : Common Objects in Context*. Dans European Conference on Computer Vision, 2014.

[Liu 03] Tianming Liu, Hong-Jiang Zhang et Feihu Qi. *A novel video key-frame-extraction algorithm based on perceived motion energy model*. *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

[Liu 16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg. *SSD : Single shot multibox detector*. Dans European Conference on Computer Vision, 2016.

[Lowe 04] David G Lowe. *Distinctive image features from scale-invariant keypoints*. *International Journal of Computer Vision*, 2004.

[Loy 09] Chen Change Loy, Tao Xiang et Shaogang Gong. *Multi-camera activity correlation analysis*. Dans Conference on Computer Vision and Pattern Recognition, 2009.

[MacQueen 67] James MacQueen et autres. *Some methods for classification and analysis of multivariate observations*. Dans Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[Mahapatra 13] Dwarikanath Mahapatra, Peter J Schüffler, Jeroen AW Tielbeek, Franciscus M Vos et Joachim M Buhmann. *Semi-supervised and active learning for automatic segmentation of Crohn's disease*. Dans International Conference on Medical Image Computing and Computer-Assisted Intervention, 2013.

[Makris 04] Dimitrios Makris, Tim Ellis et James Black. *Bridging the gaps between cameras*. Dans Conference on Computer Vision and Pattern Recognition, 2004.

[Malon 18] Thierry Malon, Geoffrey Roman-Jimenez, Patrice Guyot, Sylvie Chambon, Vincent Charvillat, Alain Crouzil, André Péninou, Julien Pinquier, Florence Sèdes et Christine Sénac. *Toulouse campus surveillance dataset : scenarios, soundtracks, synchronized videos with overlapping and disjoint views*. Dans ACM Multimedia Systems Conference, 2018.

[Malon 19a] Thierry Malon, Sylvie Chambon, Vincent Charvillat et Alain Crouzil. *Aide à la navigation dans un ensemble de vidéos par reformulation de trajectoires*. Dans Journées francophones des jeunes chercheurs en vision par ordinateur, 2019.

[Malon 19b] Thierry Malon, Sylvie Chambon, Vincent Charvillat et Alain Crouzil. *Estimation of correspondent trajectories in multiple overlapping synchronized videos using correlation of activity functions*. Dans International Conference on Image Processing, 2019.

[Malon 20a] Thierry Malon, Sylvie Chambon, Alain Crouzil et Vincent Charvillat. *Estimation des liens de recouvrement dans une collection de vidéos parcomparaison d'histoires*. Dans Congrès Reconnaissance des Formes, Image, Apprentissage et Perception, 2020.

[Malon 20b] Thierry Malon, Sylvie Chambon, Alain Crouzil et Vincent Charvillat. *Story comparison for estimating field of view overlap in a video collection.* Dans International Conference on Pattern Recognition, 2020.

[Meghdadi 13] Amir H Meghdadi et Pourang Irani. *Interactive exploration of surveillance video through action shot summarization and trajectory visualization. IEEE Transactions on Visualization and Computer Graphics*, 2013.

[Moussaïd 11] Mehdi Moussaïd, Dirk Helbing et Guy Theraulaz. *How simple rules determine pedestrian behavior and crowd disasters. Proceedings of the National Academy of Sciences*, 2011.

[Nguyen 12] Cuong Nguyen, Yuzhen Niu et Feng Liu. *Video summagator : an interface for video summarization and navigation.* Dans Conference on Human Factors in Computing Systems, 2012.

[Ofli 13] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal et Ruzena Bajcsy. *Berkeley MHAD : A comprehensive multimodal human action database.* Dans Workshop on Applications of Computer Vision, 2013.

[Oh 11] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis et autres. *A large-scale benchmark dataset for event recognition in surveillance video.* Dans Conference on Computer Vision and Pattern Recognition, 2011.

[Papageorgiou 98] Constantine P Papageorgiou, Michael Oren et Tomaso Poggio. *A general framework for object detection.* Dans International Conference on Computer Vision, 1998.

[Pritch 08] Yael Pritch, Alex Rav-Acha et Shmuel Peleg. *Nonchronological video synopsis and indexing. Pattern Analysis and Machine Intelligence*, 2008.

[Reda 18] Fitsum A Reda, Guilin Liu, Kevin J Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao et Bryan Catanzaro. *SDC-Net : Video prediction using spatially-displaced convolution.* Dans European Conference on Computer Vision, 2018.

[Redmon 16] Joseph Redmon, Santosh Divvala, Ross Girshick et Ali Farhadi. *You only look once : Unified, real-time object detection.* Dans Conference on Computer Vision and Pattern Recognition, 2016.

[Redmon 17] Joseph Redmon et Ali Farhadi. *YOLO9000 : better, faster, stronger.* Dans Conference on Computer Vision and Pattern Recognition, 2017.

[Redmon 18] Joseph Redmon et Ali Farhadi. *YOLOv3 : An incremental improvement. arXiv preprint arXiv:1804.02767*, 2018.

[Ren 15] Shaoqing Ren, Kaiming He, Ross Girshick et Jian Sun. *Faster R-CNN : Towards real-time object detection with region proposal networks.* Dans Advances in Neural Information Processing Systems, 2015.

[Ristani 16] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara et Carlo Tomasi. *Performance measures and a data set for multi-target, multi-camera tracking.* Dans European Conference on Computer Vision, 2016.

[Roman-Jimenez 20] Geoffrey Roman-Jimenez, Patrice Guyot, Thierry Malon, Sylvie Chambon, Vincent Charvillat, Alain Crouzil, André Péninou, Julien Pinquier, Florence Sedes et Christine Sénac. *Improving Vehicle Re-Identification using CNN Latent Spaces : Metrics Comparison and Track-to-track Extension. Institution of Engineering and Technology Computer Vision*, 2020.

[Rossetto 19] Luca Rossetto, Mahnaz Amiri Parian, Ralph Gasser, Ivan Giangreco, Silvan Heller et Heiko Schuldt. *Deep learning-based concept detection in vitrivr.* Dans International Conference on Multimedia Modeling, 2019.

[Ruiz-Hidalgo 12] Javier Ruiz-Hidalgo, Josep Ramon Morros, Payman Aflaki, Felipe Calderero et Ferran Marqués. *Multiview depth coding based on combined color/depth segmentation. Journal of Visual Communication and Image Representation*, 2012.

[Russakovsky 15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein et autres. *Imagenet large scale visual recognition challenge. International Journal of Computer Vision*, 2015.

[Russell 06] David Mark Russell et Shaogang Gong. *Minimum Cuts of A Time-Varying Background.* Dans British Machine Vision Conference, 2006.

[Sabata 18] Tomás Sabata, Petr Pulc et Martin Holena. *Semi-supervised and Active Learning in Video Scene Classification from Statistical Features. European Conference on Principles of Data Mining and Knowledge Discovery*, 2018.

[Schaefer 11] Gerald Schaefer. *Interactive navigation of image collections.* Dans International Conference on Future Generation Information Technology, 2011.

[Scheffer 01] Tobias Scheffer, Christian Decomain et Stefan Wrobel. *Active hidden Markov models for information extraction.* Dans International Symposium on Intelligent Data Analysis, 2001.

[Settles 08] Burr Settles, Mark Craven et Soumya Ray. *Multiple-instance active learning.* Dans Advances in Neural Information Processing Systems, 2008.

[Settles 09] Burr Settles. Active learning literature survey. Rapport Technique, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[Seung 92] H Sebastian Seung, Manfred Opper et Haim Sompolinsky. *Query by committee.* Dans Workshop on Computational Learning Theory, 1992.

[Shahroudy 16] Amir Shahroudy, Jun Liu, Tian-Tsong Ng et Gang Wang. *NTU RGB+ D : A large scale dataset for 3D human activity analysis.* Dans Conference on Computer Vision and Pattern Recognition, 2016.

[Singh 10] Sanchit SINGH, Sergio A VELASTIN et Hossein RAGHEB. *MuHaVi : A multicamera human action video dataset for the evaluation of action recognition methods.* Dans International Conference on Advanced Video and Signal Based Surveillance, 2010.

[Sinha 19] Samarth SINHA, Sayna EBRAHIMI et Trevor DARRELL. *Variational adversarial active learning.* Dans Proceedings of the IEEE International Conference on Computer Vision, pages 5972–5981, 2019.

[Stauffer 03] Chris STAUFFER et Kinh TIEU. *Automated multi-camera planar tracking correspondence modeling.* Dans Conference on Computer Vision and Pattern Recognition, 2003.

[Sunderrajan 15] Santhoshkumar SUNDERRAJAN et BS MANJUNATH. *Context-aware hypergraph modeling for re-identification and summarization. IEEE Transactions on Multimedia*, 2015.

[Van Den Hengel 06] Anton VAN DEN HENGEL, Anthony DICK et Rhys HILL. *Activity topology estimation for large networks of cameras.* Dans International Conference on Video and Signal Based Surveillance, 2006.

[Varior 16] Rahul Rama VARIOR, Mrinal HALOI et Gang WANG. *Gated siamese convolutional neural network architecture for human re-identification.* Dans European Conference on Computer Vision, 2016.

[Vezzani 08] Roberto VEZZANI et Rita CUCCHIARA. *ViSOR : Video surveillance on-line repository for annotation retrieval.* Dans International Conference on Multimedia and Expo, 2008.

[Vezzani 10] Roberto VEZZANI et Rita CUCCHIARA. *Video surveillance online repository (ViSOR) : an integrated framework. Multimedia Tools and Applications*, 2010.

[Viola 01] Paul VIOLA et Michael JONES. *Rapid object detection using a boosted cascade of simple features.* Dans Conference on Computer Vision and Pattern Recognition, 2001.

[Viola 04] Paul VIOLA et Michael J JONES. *Robust real-time face detection. International Journal of Computer Vision*, 2004.

[Vishwakarma 13] Sarvesh VISHWAKARMA et Anupam AGRAWAL. *A survey on activity recognition and behavior understanding in video surveillance. The Visual Computer*, 2013.

[Wang 14] Jiang WANG, Xiaohan NIE, Yin XIA, Ying WU et Song-Chun ZHU. *Cross-view action modeling, learning and recognition.* Dans Conference on Computer Vision and Pattern Recognition, 2014.

[Weinland 06] Daniel WEINLAND, Remi RONFARD et Edmond BOYER. *Free viewpoint action recognition using motion history optvolumes. Computer Vision and Image Understanding*, 2006.

[Yang 14] Yang Yang, Jimei Yang, Junjie Yan, Shengcai Liao, Dong Yi et Stan Z Li. *Salient color names for person re-identification.* Dans European Conference on Computer Vision, 2014.

[Zhang 06] Zhang Zhang, Kaiqi Huang et Tieniu Tan. *Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes.* Dans International Conference on Pattern Recognition, 2006.

[Zhang 19] Qi Zhang et Antoni B Chan. *Wide-area crowd counting via ground-plane density maps and multi-view fusion cnns.* Dans Conference on Computer Vision and Pattern Recognition, 2019.

[Zhu 05] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Rapport Technique, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[Zhu 19] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao et Bryan Catanzaro. *Improving semantic segmentation via video propagation and label relaxation.* Dans Conference on Computer Vision and Pattern Recognition, 2019.

Annexe A

Projet VICTORIA

Le consortium du projet VICTORIA se compose de quatorze partenaires issus de sept pays européens, parmi lesquels quatre ministères de l'intérieur, six groupes de recherche, une société de gestion de l'innovation, une PME et deux compagnies industrielles majeures dans le domaine de la sécurité. Le projet se compose de différents lots (*« work packages »*) constitués de tâches auxquelles sont associées un ou plusieurs partenaires. Les lots 1 à 3 concernent respectivement la gestion du projet, la spécification de scénarios de référence, et les aspects éthiques et légaux. Les lots 4 à 7 concernent les aspects techniques, à savoir le développement d'un logiciel destiné à être utilisé par des policiers et disposant de différentes fonctionnalités en termes d'analyse audio (lot 5) et d'analyse vidéo (lot 6). C'est sur ce dernier que nous avons eu l'occasion d'intervenir dans le cadre du projet. Le lot 8 prévoit une formation des futurs utilisateurs issus de la police à l'emploi du logiciel. Enfin, les lots 9 et 10 concernant l'organisation de démonstrations du logiciel, la diffusion et la communication autour du projet. La figure A.1 présente les interactions entre les différents lots.

Dans le cadre du projet, des vidéos ont été réalisées et mettent en scène six scénarios d'enquête filmés depuis une ou plusieurs caméras :

1. des allées et venues dans un couloir ;
2. des passages de piétons et de véhicules dans un parking ;
3. une attaque à la bombe sur une place publique ;
4. une attaque dans un bus ;
5. des personnes sur le toit d'un bâtiment ;
6. des passages de véhicules sur une route.

De nombreux acteurs professionnels ont été impliqués dans ces scénarios. En juillet 2020, suite à la rétractation d'une des actrices, la totalité des vidéos où elle apparaît — notamment la totalité du scénario 4 — ont dû être supprimées. Dans le cadre du lot concernant les dispositions éthiques et légales, l'utilisation de ces vidéos nécessitait de tenir un historique daté recensant l'intégralité des traitements appliqués. D'autre part, la publication

d'images issues de ces vidéos nécessite une autorisation. Enfin, peu de données se sont révélées exploitables dans le cadre de nos approches : la plupart des caméras sont mobiles et seules quelques vidéos durent plus d'une minute. En raison de ces procédures lourdes et du manque de données exploitables, nous avons décidé de ne pas utiliser ces vidéos.

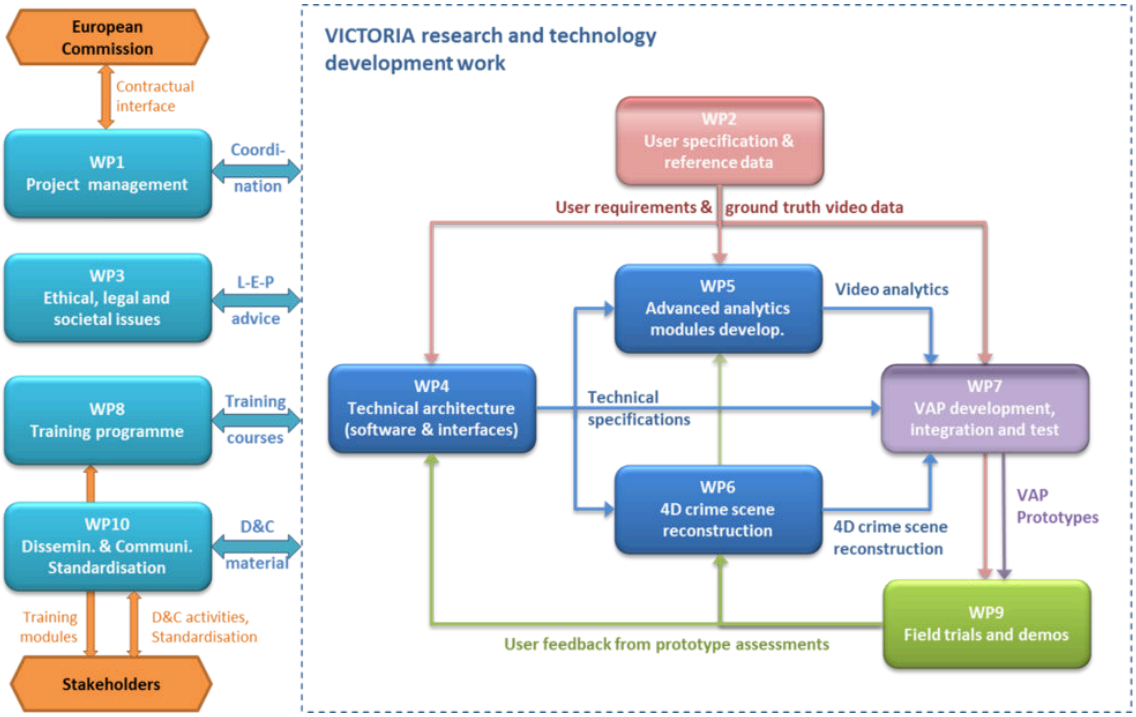


FIGURE A.1 – Interactions entre lots du projet VICTORIA — Figure issue du site Web du projet https://www.victoria-project.eu/about-victoria/.

Annexe B

Vision par ordinateur

B.1 Homographie

Considérons deux caméras C_1 et C_2 dont les champs de vue s'intersectent et $n \geqslant 4$ points coplanaires observés par ces deux caméras. Notons $(\mathbf{p}_1, ..., \mathbf{p}_n)$ (respectivement $(\mathbf{q}_1, \ldots, \mathbf{q}_n)$) les coordonnées des projetés de ces n points dans le plan image de C_1 (respectivement C_2). Les points $(\mathbf{p}_1, ..., \mathbf{p}_n)$ (respectivement $(\mathbf{q}_1, \ldots, \mathbf{q}_n)$) étant les projetés de points coplanaires, ils forment un plan projectif que l'on notera π_1 (respectivement π_2). Les points sont considérés en **coordonnées homogènes** : ils disposent d'une troisième coordonnée permettant les calculs matriciels en géométrie projective. À partir des correspondances entre ces n paires de points, il est possible d'estimer les paramètres d'une transformation linéaire, nommée **homographie**, entre π_1 et π_2, c'est-à-dire que tout point $\mathbf{p}_i$ du plan image de C_1 est lié à son correspondant $\mathbf{q}_i$ dans le plan image de C_2 par la relation $\mathbf{p}_i \sim \mathbf{H}\mathbf{q}_i$. Le symbole $\sim$ indique l'égalité à un facteur multiplicatif près. On obtient alors les coordonnées des points $(\mathbf{q}_1, ..., \mathbf{q}_n)$ dans le plan image en divisant, en coordonnées homogènes, leurs deux premières composantes par la troisième.

La figure B.1 illustre la correspondance entre points à partir d'une homographie.

FIGURE B.1 – Une homographie entre deux ensembles de points — Les quatre coins d'une place de parking sont visibles dans les vues des deux caméras. L'homographie est estimée à partir des correspondances entre les quatre coins de la place de parking, représentés par des croix noires. Pour un autre point de la vue de gauche, en violet, l'homographie permet alors d'obtenir automatiquement son correspondant dans la vue de droite, en bleu.

B.2 Critères d'évaluation

Dans les deux cas, l'indicateur le plus couramment utilisé pour évaluer les performances d'un algorithme de détection d'objets est l'indice de Jaccard, aussi appelé **l'intersection sur l'union** (*IoU*). Il est défini comme le rapport entre, d'une part, l'intersection de la boîte englobante de référence (tracée manuellement par un humain) avec la boîte englobante prédite par l'algorithme, et d'autre part, l'union de ces deux mêmes boîtes englobantes. La figure B.2 illustre ce calcul.

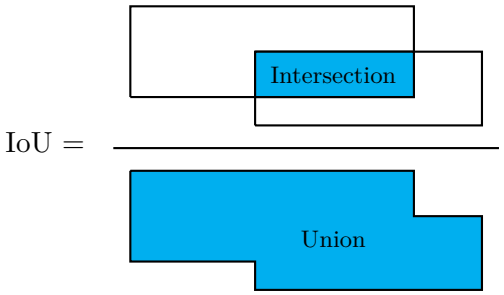


FIGURE B.2 – Calcul de l'indice de Jaccard – Définition de l'indice de Jaccard de deux boîtes englobantes qui consiste à calculer le rapport entre l'intersection et l'union de deux ensembles, d'où son appellation intersection sur union (IoU).

Ce rapport est compris entre 0 et 1. Il est d'autant plus proche de 1 que les deux boîtes englobantes coïncident, et d'autant plus proche de 0 que les deux boîtes se recouvrent peu proportionnellement à l'aire de leur union. La figure B.3 présente quelques exemples de valeurs d'IoU pour des paires de boîtes englobantes. On considère typiquement que la détection est correcte lorsque l'indice de Jaccard est supérieur à 0,5.

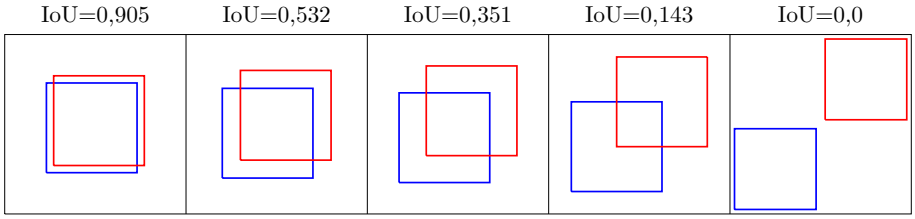


FIGURE B.3 – Quelques exemples d'indices de Jaccard – Plus les boîtes englobantes coïncident, plus l'union est proche de l'intersection et donc plus l'IoU est proche de 1.

Pour évaluer les performances d'un algorithme de détection d'objets, il existe des jeux de données largement utilisés dans la communauté scientifique qui contiennent de nombreuses images d'objets de différentes catégories, accompagnées des annotations des boîtes englobantes de référence que l'on qualifie communément de « vérité terrain ». Parmi les plus célèbres d'entre eux, il y a le *Pascal Visual Object Classes Dataset* (Pascal VOC Dataset) [Everingham 10] constitué de 11 530 images contenant 27 450 objets annotés appartenant à 20 catégories. Chaque année se tient le *Pascal VOC Challenge* au cours duquel les

approches des candidats sont classées en fonction du pourcentage d'objets de l'ensemble de test qu'elles détectent avec un indice de Jaccard supérieur à un seuil de 0,5. Cette compétition a été la première à proposer un jeu de données aussi riche accompagné d'une méthode d'évaluation. Elle a posé les bases de l'évaluation en détection d'objets.

Plus récemment, en 2015, *Imagenet Large Scale Visual Recognition Challenge* (ILSVRC) [Russakovsky 15], est devenu la compétition de référence. En s'appuyant sur le jeu de données Imagenet [Deng 09] qui contient près de 1,5 million d'images de 1 000 catégories différentes, cette compétition propose le même type d'évaluation que Pascal VOC pour la détection d'objets, à la différence près que le seuil d'indice de Jaccard à atteindre n'est pas pris uniformément égal à 0,5, en particulier pour les objets de petite taille. En effet, [Russakovsky 15] estiment que l'erreur moyenne d'annotation est de l'ordre de 5 pixels sur chacune des deux dimensions de l'image. Par exemple, pour un objet de taille 10×10 pixels, cela représente une boîte englobante de taille 20×20, soit un indice de Jaccard égal à $100/400 = 0{,}25$, alors même qu'il s'agit d'une annotation manuelle. Pour favoriser la prise en compte des détections d'objets de petite taille, les auteurs ont donc proposé un seuil qui dépend des dimensions (w, h) de la boîte englobante B, pris égal à :

$$\text{seuil}(B) = \min\left(0{,}5, \frac{wh}{(10+w)(10+h)}\right).$$

Enfin, un autre jeu d'images largement utilisé est le *Common objects in context dataset* (COCO dataset) [Lin 14]. Il contient près de 330 000 images sur lesquelles ont été annotés 2,5 millions d'objets de 91 catégories différentes. Contrairement aux deux bases d'images précédentes qui ne contiennent que des annotations sous forme de boîtes englobantes, COCO fournit aussi la segmentation de chaque objet.

B.3 Méthodes de comparaison et de classification des descripteurs

Dans une approche de détection, le descripteur associé à une région de l'image est comparé à un modèle d'objet d'une certaine catégorie pour déterminer si la région contient ou non un objet de cette catégorie. Ce modèle est appris sur un ensemble d'images d'objets de la catégorie. Pour déterminer la présence et la catégorie d'un objet dans une image donnée, on calcule son descripteur et on le soumet à un classifieur. Ce dernier a été préalablement entraîné de façon supervisée, c'est-à-dire qu'on lui a donné un ensemble de paires (descripteur, classe), le descripteur ayant été calculé à partir d'une image, et la classe correspondant à la catégorie de l'objet présent sur l'image. Les séparateurs à vaste marge (SVM) [Boser 92], présentés au chapitre 4 sont un exemple de classifieur simple.

Dans une approche de réidentification, les descripteurs sont comparés entre eux. Diffé-

rentes mesures de distance entre deux vecteurs descripteurs $\mathbf{D}_1$ et $\mathbf{D}_2$ ont été proposées. Parmi les plus classiques, nous pouvons citer la distance L_2 :

$$\mathrm{d}_{L_2}(\mathbf{D}_1, \mathbf{D}_2) = ||\mathbf{D}_1 - \mathbf{D}_2||. \tag{B.1}$$

La distance cosinus est également très utilisée :

$$\mathrm{d}_{cosinus}(\mathbf{D}_1, \mathbf{D}_2) = \frac{\mathbf{D}_1^{\mathrm{T}}\mathbf{D}_2}{||\mathbf{D}_1||\,||\mathbf{D}_2||}. \tag{B.2}$$

En réidentification, la mesure KISSME (*Keep It Simple And Straightfoward MEtric*) [Koestinger 12] est aussi régulièrement utilisée. Elle fait appel à la distance de Mahalanobis qui fait intervenir des matrices de covariance calculées sur les paires d'objets similaires et sur les paires d'objets dissemblables.

$$\mathrm{d}_{Mahalanobis}(\mathbf{D}_1, \mathbf{D}_2) = (\mathbf{D}_1 - \mathbf{D}_2)^{\mathrm{T}}\mathbf{M}(\mathbf{D}_1 - \mathbf{D}_2) \tag{B.3}$$

où $\mathbf{M} = \mathbf{M}_S^{-1} - \mathbf{M}_D^{-1}$ et $\mathbf{M}_S$ (respectivement $\mathbf{M}_D$) sont les matrices de covariance des paires d'images similaires (respectivement dissemblables).

Annexe C

Aspects techniques

C.1 Squelette d'interface interactive en `Javascript`

Au cours de cette thèse, nous avons été amenés à concevoir plusieurs prototypes d'interfaces interactives. Dans ce but, nous avons choisi d'utiliser le langage `Javascript`. Il présente en effet les avantages suivants :

— sa synthaxe est simple et les fonctionnalités graphiques nombreuses, ce qui permet de réaliser des prototypes rapidement ;

— les navigateurs Web peuvent exécuter directement du code `Javascript`, il est facile de mettre ses programmes en ligne sous la forme de pages HTML ;

— les moteurs `Javascript` récents sont très optimisés par rapport aux autres langages similaires, par exemple `Python`.

Nous décrivons ici le squelette commun aux différentes interfaces codées pendant la thèse. Nous présentons également quelques astuces d'affichage que nous avons mises en place au cours de cette thèse et qui nous ont semblé pertinentes.

Le squelette du programme se compose d'une fonction d'initialisation qui définit un certain nombre de variables et d'une fonction qui se rappelle elle-même indéfiniment. Cette dernière fonction est en charge de mettre à jour l'affichage à chaque itération, près de 60 fois par seconde. Elle commence pour cela par effacer le contenu affiché, puis redessine tous les éléments. Nous détaillens plus loin les fonctions d'affichage les plus courantes.

La fonction d'initialisation définit le contexte, qui est l'objet qui permet l'affichage. Elle définit également des variables relatives aux coordonnées de la souris et des récepteurs d'évènements (*event listeners*) qui permettent de détecter les mouvements et les clics de souris. On dispose ainsi en permanence des coordonnées du pointeur de la souris et du dernier clic. Les autres variables utiles, telles que les tableaux d'éléments, potentiellement vides initialement, sont aussi à définir dans cette fonction, ainsi que les images.

```javascript
// interface.js
initialiser()
animer()

function initialiser() {
    // Récupérer l'objet canevas et son contexte
    canvas = document.getElementById("canvas")
    ctx = canvas.getContext("2d")

    // Initialiser une variable pour stocker les données relatives à la souris
    souris = {
        "pos": {"x": -1, "y": -1},
        "clic": {"x": -1, "y": -1},
        "dernier_clic": 0
    }

    // Pour récupérer les coordonnées de la souris quand elle bouge
    canvas.addEventListener('mousemove', function(event) {
        event.preventDefault()
        souris.pos = getMousePos(canvas, event)
    }, false)

    // Pour récupérer les coordonnées de la souris quand on clique
    canvas.addEventListener('mousedown', function(event) {
        event.preventDefault()
        souris.clic = getMousePos(canvas, event)
        souris.dernier_clic = event.which // vaut 1 pour un clic gauche et 3
            pour un clic droit
    }, false)
}

function animer() {
    // On raffraîchit la fenêtre
    ctx.clearRect(0, 0, canvas.width, canvas.height)

    // Corps de boucle : affichages, mises à jour...
    ...

    // Pour rappeler cette fonction 60 fois par seconde
    requestAnimationFrame(animer)
}
```

Snippet C.1 – Squelette d'interface en `Javascript`.

La fonction `getMousePos` permettant de récupérer un objet disposant de deux attributs `x` et `y` correspondant à la position du pointeur de la souris dans la fenêtre s'écrit :

```
function getMousePos(c, event) {
    var rect = c.getBoundingClientRect()
    return {
        "x": event.clientX - rect.left,
        "y": event.clientY - rect.top
    }
}
```

Snippet C.2 – Fonction de récupération des coordonnées du pointeur de la souris en `Javascript`.

Dans le corps de la boucle, les fonctionnalités les plus utiles à l'affichage sont le tracé de segments et de cercles, l'affichage de texte et l'affichage d'image. En dehors de l'affichage d'image, toutes les fonctions citées utilisent trois paramètres propres à l'objet contexte : deux paramètres de couleur, respectivement `strokeStyle` pour le contour et `fillStyle` pour l'intérieur, et un paramètre d'épaisseur du contour `lineWidth`.

Pour tracer un arc de cercle de centre `(x, y)` et de rayon `r`, on peut utiliser les fonctions suivantes (pour un cercle complet, on utilise `angle_initial = 0` et `angle_final = 2 * Math.PI`) :

```
ctx.beginPath()
ctx.arc(x, y, r, angle_initial, angle_final)
ctx.stroke() // Tracé du contour
ctx.fill()  // Pour remplir l'intérieur du cercle
ctx.endPath()
```

Snippet C.3 – Tracé d'un arc de cercle en `Javascript`.

Pour tracer un segment entre deux points de coordonnées respectives `(x1, y1)` et `(x2, y2)`, on peut utiliser les instructions suivantes :

```
ctx.beginPath()
ctx.moveTo(x1, y1)
ctx.lineTo(x2, y2)
ctx.stroke()
ctx.endPath()
```

Snippet C.4 – Tracé d'un segment en `Javascript`.

Il est également possible de tracer une forme quelconque à partir d'une succession de `lineTo`. Dans ce cas, l'intérieur de la forme peut être remplie en utilisant `ctx.fill()` comme dans le cas du cercle.

Pour afficher du texte et sélectionner sa police et sa taille, on utilise les instructions suivantes :

```
ctx.font = "20px Arial" // Pour une police de caractères Arial en taille 20
ctx.strokeText(chaine_de_caracteres, x, y) // Pour tracer le contour du texte
ctx.fillText(chaine_de_caracteres, x, y) // Pour remplir l'intérieur du texte
```

Snippet C.5 – Affichage de texte et choix de la police et de sa taille en `Javascript`.

Enfin, la définition d'une image et son affichage se font par les instructions suivantes :

```
image = new Image()
image.src = chemin_vers_l_image
ctx.drawImage(image, x, y, largeur, hauteur)
```

Snippet C.6 – Définition et affichage d'une image en `Javascript`.

Les deux premiers paramètres, `x` et `y`, correspondent à la position du coin en haut à gauche de l'image affichée. Si l'on souhaite donner les coordonnées du centre de l'image, on peut faire :

```
ctx.drawImage(image, x - largeur / 2, y - hauteur / 2, largeur, hauteur)
```

Snippet C.7 – Affichage d'une image centrée en `(x, y)` en `Javascript`.

Une autre syntaxe de la fonction `drawImage` permet de sélectionner la zone de l'image que l'on souhaite afficher, délimitée par un rectangle obtenu au travers de 4 paramètres supplémentaires.

C.2 Astuces d'affichage en `Javascript`

Lorsque des éléments changent de position dans une interface, il y a un risque que l'utilisateur perde ses repères, notamment si l'affichage change instantanément d'une image à l'autre. Pour éviter cela, nous assignons à chaque élément deux attributs par coordonnées de position : une cordonnée réelle qui correspond à l'endroit où est supposé se situer l'élément et une coordonnée à laquelle l'élément est effectivement affiché. Cette dernière se met à jour en permanence pour se rapprocher d'un facteur α de la distance qui la sépare de la coordonnée réelle :

```
élément.x_affiché = (1 - alpha) * élément.x_affiché + alpha * élément.x_réel
élément.y_affiché = (1 - alpha) * élément.y_affiché + alpha * élément.y_réel
```

Snippet C.8 – Mise à jour dynamique de la position d'un élément en `Javascript`.

Pour mettre en évidence un élément, par exemple un élément cliqué, il est possible de le faire clignoter en modifiant sa transparence au cours du temps selon une période T. Nous utilisons pour cela les lignes suivantes :

```
t = new Date().getTime() // Nombre de millisecondes depuis le 1er janvier 1970
// Pour une variation linéaire continue entre 0 et 1 :
ctx.globalAlpha = Math.abs( (t % T) - T / 2 ) / ( T / 2 )
affichage(élément)
ctx.globalAlpha = 1
```

Snippet C.9 – Changement de la transparence pour faire clignoter un élément en `Javascript`.