Dynamic visual saliency in image sequences

Léo Maczyta

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

École Doctorale N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, vision*

Par

Léo MACZYTA

« Dynamic visual saliency in image sequences »

Thèse présentée et soutenue à RENNES, le 25 Novembre 2020
Unité de recherche : Serpico, Inria Rennes Bretagne Atlantique
Thèse N° :

Rapporteurs avant soutenance :

Alice Caplier, Professeur, Grenoble INP

Thierry Chateau, Professeur, Université Clermont-Auvergne

Composition du jury :

Attention, en cas dabsence dun des membres du Jury le jour de la soutenance, la composition ne comprend que les membres présents

Président :	Prénom Nom Fonction et établissement d'exercice
Examinateurs :	Alice Caplier, Professeur, Grenoble INP
	Thierry Chateau, Professeur, Université Clermont-Auvergne
	Elisa Fromont, Professeur, Université de Rennes 1
	Patrick Pérez, DR INRIA - Directeur scientifique de Valeo.ai
	Véronique Serfaty, Responsable innovation pôle numérique, AID/DGA
Dir. de thèse :	Patrick Bouthemy, DR INRIA, Centre INRIA Rennes-Bretagne Atlantique
Co-dir. de thèse :	Olivier Le Meur, Professeur, Université de Rennes 1

REMERCIEMENTS

Je tiens à remercier Patrick Bouthemy et Olivier Le Meur pour m'avoir proposé d'étudier la saillance du mouvement, et de m'avoir accompagné et guidé pendant cette thèse. Ce sujet est lié à de multiples problématiques pratiques et théoriques, qui ont connu des évolutions importantes ces dernières années et auxquelles se rattachent de nombreuses questions encore ouvertes. Pouvoir les étudier et expérimenter autour de ces aspects m'a permis de beaucoup apprendre.

I want to thank the members of the Serpico and Fluminance teams, past and present, who were present along this journey. Thanks to all of you.

Je remercie également Huguette Béchu, pour son aide pour les aspects pratiques, y compris pendant cette période particulière.

Une mention spéciale aux igrida, yupana et autres abacus, toujours prêts à éclairer les sentiers de la connaissance.

Enfin, merci à ma famille pour tout ce qu'elle m'a apporté. Puisse le pigeon voyageur qui vous portera ce manuscrit trouver son chemin.

Financement

Cette thèse a été cofinancée par la DGA et la région Bretagne.

TABLE OF CONTENTS

RÉSUMÉ EN FRANÇAIS

Contexte et motivations

Afin de guider nos actions dans l'environnement dans lequel nous évoluons, nous disposons de nos sens, classiquement la vision, l'audition, le toucher, le goût et l'odorat. D'autres pourraient être ajoutés à cette liste, tels que le sens de l'équilibre ou la capacité à ressentir la température. Certains animaux disposent d'ailleurs de sens qui nous sont inaccessibles, comme la magnétoréception pour certains cétacés et oiseaux, qui leur permet de percevoir les champs magnétiques, ou encore l'écholocalisation pour les chauve-souris ou les dauphins notamment, qui leur sert à sonder l'espace autour d'eux. Des capteurs complexes et variés sont à l'origine de ces sens. Parmi ceux qui ont été énumérés, la vision présente un intérêt particulier. En effet, dans un environnement constitué d'air, de vide ou même d'eau peu profonde, la vision fournit une information riche et détaillée sur les objets environnants. De plus, étant fondée sur la perception de la lumière, la vision permet d'obtenir des informations sur des objets distants pratiquement instantanément. Pour cette raison, nous avons tendance à largement nous reposer sur ce sens pour analyser notre environnement et y évoluer.

Même en se concentrant uniquement sur la vision, l'environnement reste particulièrement complexe. Il est commun de voir simultanément de nombreux objets et personnes, chacun d'entre eux requérant potentiellement une action de la part de l'observateur. Une première possibilité pour correctement évoluer dans de tels environnements serait de prêter attention en permanence à chaque élément de la scène. Bien qu'une approche de ce type conviendrait en théorie, ce n'est pas ce que nous faisons en pratique. Tout d'abord, notre champ de vision est limité. À l'intérieur de ce champ, notre vision centrale, qui s'appuie sur la partie de l'œil appelée la fovéa, est plus précise que notre vision périphérique. Ce fait implique qu'il est nécessaire de bouger la tête et les yeux pour pouvoir analyser l'ensemble des régions de l'environnement, et en particulier pour pouvoir se focaliser sur un élément. De plus, une analyse à la fois permanente et détaillée de la scène observée requerrait un coût en énergie plus élevé, sans compter la nécessité d'une plus grande complexité du cerveau pour pouvoir mener cette analyse.

Pour pouvoir analyser en permanence notre environnement à un coût raisonnable, nous nous appuyons sur l'attention visuelle. L'attention visuelle est contrôlée à la fois par le monde extérieur et par le cerveau. Son rôle est de sélectionner des information importantes en s'appuyant sur les indices de saillance. Une fois que les éléments d'intérêt sont identifiés, il est possible de nous focaliser sur eux et d'allouer nos ressources à leur analyse.

Une première famille d'indices de saillance est liée à l'apparence des objets. Par exemple, une poignée de framboises rouges sera perçue comme saillante sur un fond de buissons, d'arbres et d'herbe verts. Dans une scène avec une majorité d'objets ayant une apparence similaire, il est aisé de prendre conscience de la présence de ces objets. Il n'y a pas de raison particulière de regarder un objet plutôt qu'un autre. Par contre, s'il y a un objet qui a une apparence très différente, le réflexe sera certainement de l'analyser plus en détail et de se focaliser sur celui-ci. Ce type d'indices de saillance relève essentiellement d'une comparaison avec une apparence normale dans la scène.

FIGURE 1 – Exemples de cas où la saillance est due à des indices d'apparence (première ligne) et de mouvement (seconde ligne). Le cheval et le panneau sont clairement distincts de leur environnement, ce qui est bien entendu voulu pour le panneau. Les deux caméléons dans l'image en bas à gauche sont nettement plus faciles à distinguer lorsqu'ils sont vus en train de bouger. Si un piéton traversait la route pour la configuration illustrée en bas à droite, le mouvement des voitures serait un élément central auquel prêter attention. Les images proviennent de MSRA-B [Wan+17], Camouflaged Animals [BLM16] et de FBMS-59 [OMB14].

Une deuxième famille d'indices de saillance est liée au mouvement. Face à une scène statique, seuls quelques instants sont généralement suffisants pour avoir un aperçu de la scène. Ensuite, si rien n'a bougé, il est possible d'en déduire qu'il n'y a pas de nouveaux éléments et que l'analyse passée reste valable. Au contraire, si un objet en mouvement entre dans le champ de vision, cet objet n'a naturellement pas été analysé et mérite plus d'attention que le reste de la scène. De façon plus générale, n'importe quel objet en mouvement de l'environnement présente un intérêt. À cause de son mouvement, il pourrait interagir avec l'observateur, en

l'approchant, le fuyant, le chassant, ou simplement en croisant son chemin. Dans toutes ces situations, l'observateur devrait réagir de façon appropriée. Les indices de mouvement sont donc pertinents pour sélectionner l'élément sur lequel se concentrer. Dans le cas où plusieurs objets sont en mouvement, la plus forte emphase devrait correspondre à l'objet avec le mouvement le plus singulier, qui peut être l'objet le plus rapide, celui suivant un mouvement particulier ou encore celui suivant un mouvement erratique, et qui est le plus susceptible de nécessiter une réponse spécifique. Les indices de mouvement sont dérivés de l'évolution de la scène au cours du temps. Naturellement, si la tête ou les yeux de l'observateur bougent, un mouvement apparent sera induit dans la scène. L'introduction dans le processus de perception de mécanismes pour prendre ce phénomène en compte devrait ainsi conduire à une meilleure estimation de la saillance.

L'apparence et le mouvement sont donc à l'origine de deux types d'indices de saillance, à la fois utiles mais distincts, comme illustré en Figure 1. L'apparence attire l'attention sur les éléments ayant les formes ou couleurs les plus singulières. Les indices de mouvement orientent l'attention vers les éléments mobiles suivant un mouvement particulier ou effectuant une action singulière.

La discussion ci-dessus s'est concentrée sur le rôle spécifique de la vision pour la perception de l'environnement des personnes et de nombreuses espèces d'animaux, ainsi que sur l'importance de la saillance pour permettre un traitement efficace du signal. Si nous nous intéressons maintenant au problème de la construction de systèmes robotiques capables d'agir de façon autonome dans leur environnement, nous pouvons constater que ces systèmes doivent conduire une analyse similaire de la scène dans laquelle ils se trouvent pour mener à bien leurs tâches. De plus, de tels systèmes ont habituellement accès à des ressources embarquées limitées et doivent respecter certaines contraintes, en particulier en ce qui concerne la consommation d'énergie, le temps de calcul ou la charge du processeur, ce qui rend un traitement efficace nécessaire. Il est également possible de considérer le problème plus général de l'analyse de vidéo, qui inclut la vidéosurveillance, le résumé de vidéos ou encore l'étude du mouvement des foules. Pour l'ensemble de ces cas, l'analyse devrait être restreinte à la partie la plus pertinente pour éviter de mobiliser inutilement des ressources de calcul et pour extraire les informations vraiment pertinentes. L'intérêt des indices de mouvement pour ces problèmes est dû au fait qu'ils fournissent des informations sur la dynamique des objets dans la scène, ce qui les rend adéquats pour permettre une navigation sûre, ou pour surveiller et détecter des comportements inhabituels et potentiellement dangereux. L'objectif de cette thèse est précisément d'étudier le problème d'estimation de la saillance du mouvement dans des vidéos.

Travaux en lien avec la saillance du mouvement

Une revue détaillée des travaux existants en lien avec la saillance du mouvement est présentée dans un chapitre dédié. Dans ce qui suit, nous nous bornerons à présenter le contexte général et les points les plus notables.

L'estimation de la saillance est un sujet de recherche actif, pouvant correspondre à un mécanisme de pré-attention ou pouvant être un objectif en soi. Lorsqu'il s'agit de traiter des images ou des vidéos, la sortie va généralement consister en des cartes de saillance fournissant la probabilité de saillance pour chaque pixel. L'estimation de la saillance est liée à l'attention visuelle, à la détection d'objets ou encore à la segmentation d'image. Il n'existe pas une unique définition formelle de la saillance d'images, qui peut se décliner de plusieurs façons. Nous avons déjà exposé la distinction entre la saillance d'apparence et de mouvement. Les déclinaisons suivantes de la saillance sont communément utilisées. Une première possibilité est de considérer que ce qui attire l'attention d'un observateur humain est saillant. Il est possible de l'enregistrer avec des appareils spécifiques de suivi du regard, dont le rôle est de mesurer les endroits où les personnes fixent leur attention. Une deuxième possibilité est de considérer que les objets avec une apparence ou un mouvement particulièrement singuliers sont saillants. Typiquement, les objets situés au premier plan qui suivent un mouvement marqué sont considérés comme saillants.

Lorsque la saillance est définie de l'une de ces façons, l'apparence et le mouvement jouent tous deux un rôle pour l'identification des éléments d'intérêt. Les méthodes conçues pour ce type de tâches ont donc naturellement tendance à s'appuyer sur ces deux aspects. En ce qui concerne l'apparence, les information sur la couleur, la texture, les contours ou les formes sont typiquement utilisées. En ce qui concerne le mouvement, la cohérence temporelle, le flot optique, à partir duquel l'amplitude de mouvement ou les frontières de mouvement peuvent être extraites, sont communément employés. La différence entre les méthodes existantes résidera principalement dans la façon dont ces éléments sont extraits et combinés pour estimer la saillance. Ceci peut se faire en concevant un algorithme dédié pour assembler et traiter ces différentes informations. Une autre possibilité est de s'appuyer sur les réseaux de neurones profonds, qui ont montré leur efficacité pour l'analyse et le traitement d'images et qui sont adoptés par la plupart des méthodes récentes.

Plus généralement, il est possible qu'un élément soit considéré comme saillant à partir du moment où il dévie de la norme ou si son comportement est inattendu. Dans les contextes ou cette définition est utilisée, il est plus commun de parler de détection d'anomalie. La détection d'anomalie peut s'appliquer sur des images ou des vidéos, mais également sur d'autres types de données. Les trajectoires peuvent ainsi être concernées, tout en étant pertinentes pour notre problème, étant donné qu'elles traduisent une information sur le mouvement avec la

particularité d'opérer sus un temps long. Pour les méthodes traitant cette famille de problèmes, le but est de repérer dans les données des comportements ou des motifs inhabituels. Cela peut se faire en exploitant les connaissances d'expert liée au problème particulier traité, en utilisant des outils statistiques ou encore en recourant à des réseaux de neurones profonds avec apprentissage.

Contributions de cette thèse

Les méthodes existantes tendent à mêler les indices de saillance de mouvement et d'apparence pour produire une estimation de la saillance unique. Cette approche conduit à perdre l'information de la cause de la saillance, ce qui est regrettable. Pour les exemples présentés plus haut (la navigation pour la robotique mobile, la vidéosurveillance...), l'intérêt peut ne porter que sur la saillance du mouvement, auquel cas la saillance liée à l'apparence ne devrait pas jouer le rôle d'un distracteur. Notons d'ailleurs qu'il existe des situations où l'apparence ne joue qu'un rôle très secondaire voire nul. Un exemple typique correspond au cas d'un individu marchant en contre-sens dans une foule. Des configurations de ce type peuvent se retrouver dans des applications diverses, telles que la surveillance du trafic urbain ou routier [RB19 ; BFM19], la sécurité des zones publiques accueillant des foules denses [PRBB17], l'étude de la dynamique des cellules pour la bio-imagerie [Rou+17], ou encore l'identification de conditions météorologiques se dégradant [Pap+00]. Dans tous ces cas, le mouvement peut être le seul élément permettant d'extraire des information pertinentes sur la scène observée.

Trois types d'entités peuvent être impliquées dans la caractérisation de la saillance du mouvement dans une séquence d'images :

— la scène statique mais observée avec une caméra mobile,
— les éléments mobiles saillants de la scène,
— les éléments mobiles normaux de la scène.

Puisque nous nous intéressons à la saillance du mouvement, la scène statique n'a pas lieu d'être considérée comme saillante. À l'opposé, les objets suivant un mouvement singulier sont saillants. Il est également possible de rencontrer des situations dans lesquelles plusieurs objets suivent le même mouvement. Dans ce cas, les objets saillants seront les objets qui suivent un mouvement qui se démarque de ce mouvement principal, plutôt que les objets faisant partie de ce flux. Maintenant que nous avons fait la distinction entre ces trois entités, nous pouvons noter qu'elles peuvent être perçues par une caméra qui peut être statique ou mobile. Dans ce dernier cas, un mouvement apparent sera induit dans l'ensemble de l'image, modifiant par là la perception du mouvement de son contenu. Nos travaux décrits dans les chapitres suivants seront positionnés par rapport à ces aspects.

Les contributions de cette thèse sont organisées autour de trois axes, qui permettent d'ana-

lyser de façon complémentaire la saillance du mouvement.

Le premier axe s'intéresse à la détection temporelle de la saillance du mouvement dans des séquences d'images. Le but est d'être capable de décider pour chaque image s'il y a présence de saillance du mouvement ou non. Dans ce dernier cas l'image peut être ignorée. À notre connaissance, ce problème n'a pas été étudié en tant que tel. Il constitue cependant un pré-requis nécessaire pour les méthodes de localisation de la saillance. Ces dernières font généralement l'hypothèse implicite que de la saillance est présente, ce qui n'est pas forcément le cas et peut conduire à l'estimation de cartes de saillance factices. Notre objectif est ainsi de construire un mécanisme de pré-attention qui peut être utilisé pour déclencher des traitements additionnels des images, uniquement si requis et légitime.

Le second axe traite de l'estimation de cartes valuées de saillance du mouvement (voir l'illustration en Figure 2). Lorsque la saillance du mouvement est détectée dans l'image, il est utile de localiser précisément l'élément saillant. Un système de robotique mobile autonome pourrait s'en servir pour décider s'il est nécessaire d'adapter sa trajectoire et de quelle façon le faire, et un système d'analyse de vidéos pourrait indiquer à un opérateur humain la localisation de l'élément d'intérêt.

FIGURE 2 – Illustration de cartes de saillance annotées à la main pour quatre exemples. Une première possibilité est que la scène ne contienne pas d'élément ayant un mouvement saillant. Dans ce cas, cela se traduit par une carte de saillance vide. Les deux exemples de la première ligne correspondent à une situation de ce type. Nous les avons acquis pour constituer la base de données qui sera décrite dans le chapitre 3. Au contraire, lorsque des éléments mobiles saillants sont présents, ils seront délimités dans la vérité terrain. C'est le cas des deux exemples de la seconde ligne, qui proviennent de la base de données DAVIS 2016 [Per+16]. Ici, la vérité-terrain est en fait exprimée par une carte binaire.

Le troisième axe se concentre sur l'estimation de la saillance de trajectoires. La saillance du mouvement peut n'apparaître qu'après un certain délai, et les trajectoires constituent un bon moyen de représenter des évènements d'une certaine durée, comme illustré en Figure 3. Les

trajectoires peuvent être extraites à partir de vidéos avec des systèmes de suivi (ou *tracking*). Les trajectoires ne sont cependant pas restreintes à cette configuration, et peuvent provenir de capteurs dédiés dont notamment les GPS, qui sont désormais répandus par exemple dans les voitures ou les téléphones portables. De façon plus générale, n'importe quelle série temporelle peut être vue comme une trajectoire et traitée de façon similaire. L'objectif de ce troisième axe est de définir une méthode pour estimer la saillance de trajectoires, c'est-à-dire identifier les trajectoires singulières dans un groupe de trajectoires semblables ou normales.

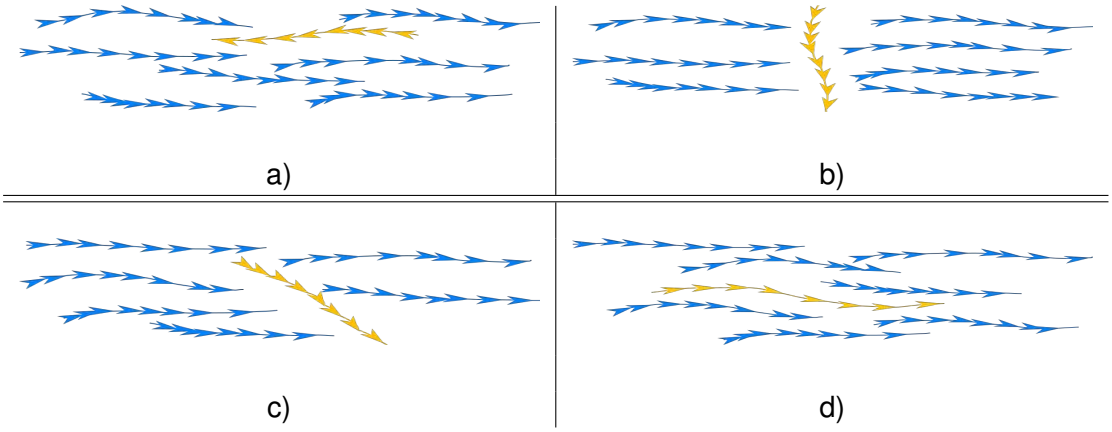


FIGURE 3 – Illustration de quatre configurations de saillance de trajectoires. Les mouvements normaux sont représentés en bleu et les mouvements saillants en orange. Dans les configurations a) et b), la trajectoire saillante se démarque clairement des trajectoires normales. Dans la configuration c), la trajectoire saillante suit une direction légèrement différente par rapport à la norme, tandis que dans la configuration d), la trajectoire saillante correspond à un mouvement plus rapide donc un parcours plus long en un temps donné. Pour ces deux derniers cas, la saillance est moins marquée. N'observer les trajectoires que pendant un bref instant risque de ne pas être suffisant pour identifier convenablement celles qui sont saillantes, en particulier si les mouvements ne sont pas parfaitement réguliers. La prise en compte d'une période de temps plus étendue autorisera une décision plus fiable.

Organisation du manuscrit

Le manuscrit est organisé en quatre chapitres principaux, en plus de l'introduction et de la conclusion générale. Le chapitre 2 consiste en une revue des travaux existants en lien avec la saillance du mouvement. Dans le chapitre 3, nous développons une méthode pour la détection de la présence de saillance du mouvement au niveau de chaque image d'une vidéo. Le chapitre 4 présente une méthode pour l'estimation de cartes valuées de saillance du mouvement fondée de manière originale sur la reconstruction ou *inpainting* du flot optique. Le chapitre 5 traite de la saillance de trajectoire. De plus, l'annexe A explore l'estimation de la saillance rela-

tive à l'aide d'une base de vidéos 3D. Ci-dessous, nous présentons un bref aperçu de chacune de ces parties et de nos contributions associées.

Chapitre 2

Dans le chapitre 2, nous passons en revue des travaux en lien avec l'estimation de la saillance et la détection d'anomalie, qui sont des domaines étroitement liés à la saillance du mouvement. Cette revue nous permet d'identifier d'autres champs de recherche pertinents pour notre objectif d'estimation de la saillance du mouvement. Plusieurs des domaines abordés dans ce chapitre seront explorés pour la définition de nos méthodes, notamment les domaines concernant les réseaux convolutifs profonds, l'estimation du flot optique, le *deep metric learning*, et l'apprentissage avec une supervision faible.

Chapitre 3

Dans le chapitre 3, nous nous intéressons au problème de la détection temporelle de la saillance du mouvement au niveau de l'image, qui consiste à prédire quelles images d'une vidéo incluent de la saillance du mouvement. La méthode que nous proposons est composée de deux étapes principales. Dans un premier temps, le mouvement d'ensemble dû au mouvement de la caméra est estimé et compensé. Dans un second temps, la classification d'une image en saillante ou non saillante est réalisée à l'aide d'un réseau convolutif. Nous avons appliqué notre méthode soit directement aux images couleur, soit au flot optique. De plus, nous avons expérimenté deux méthodes pour l'estimation du mouvement dominant de l'image dû au déplacement de la caméra. Ceci conduit à un total de quatre variantes de notre méthode, qui sont comparées et évaluées. L'apprentissage et l'évaluation expérimentale de ces variantes s'appuient sur une base de données synthétiques de paires d'images avec un mouvement connu, et une base de données de vidéos réelles. Les résultats expérimentaux montrent que la variante la plus performante, dénotée RFS-Motion2D, s'appuie sur le flot optique. La compensation du mouvement dominant est effectuée en soustrayant le flot dominant au flot optique, ce qui produit un flot résiduel utilisé comme entrée du réseau de classification. Les résultats expérimentaux montrent également que notre méthode est capable de détecter correctement la présence de saillance du mouvement, même pour des scènes dans lesquelles des objets statiques présentent un fort mouvement apparent du fait de leur position en avant-plan de la scène et du mouvement de la caméra. Nous atteignons un taux de détection correcte de 87,5% sur notre jeu de données réelles, et même de 93,3% sur la base de données DAVIS 2016 [Per+16].

Chapitre 4

Dans le chapitre 4, nous considérons le problème de l'estimation de cartes de saillance, qui consiste à prédire une valeur de saillance (codée entre 0 et 1) pour chaque pixel des images d'une vidéo. Pour le résoudre, nous définissons une méthode s'appuyant sur *l'inpainting* ou la reconstruction du flot optique. Des régions saillantes candidates sont d'abord extraites à partir des frontières du flot optique calculé. Le flot optique à l'intérieur de ces régions est ensuite reconstruit à partir du flot environnant. L'idée est que, si la région est saillante, le flot reconstruit devrait être clairement distinct du flot calculé à l'origine dans cette région. Au contraire, si la région n'est pas saillante, le flot reconstruit et le flot originel devraient être similaires. Nous exploitons la différence entre ces deux flots pour calculer la carte de saillance du mouvement. Les résultats expérimentaux montrent que notre méthode se compare favorablement à des méthodes existantes d'estimation de la saillance, en arrivant en seconde position par rapport à ces méthodes avec les métriques considérées. Nous notons que ces résultats sont obtenus sans utiliser d'indices liés à l'apparence, contrairement aux méthodes d'estimation de la saillance générique auxquelles nous comparons expérimentalement notre méthode. Ce type d'indices est en effet exploitables pour la base de test utilisée, étant donné que celle-ci n'a pas été construite spécifiquement pour l'évaluation de la saillance du mouvement. Nous n'avons également pas utilisé d'apprentissage pour l'étape d'extraction de la saillance à partir du flot optique. Nous avons ainsi défini une méthode efficace et non supervisée.

Chapitre 5

Le chapitre 5 est dédié au problème de l'estimation de la saillance de trajectoires. Les trajectoires permettent d'aborder la saillance qui n'apparaît qu'avec le temps, c'est-à-dire de manière progressive. De plus, en présence de plusieurs objets, les trajectoires de ces objets fournissent un moyen naturel de comparer leurs mouvements respectifs et d'identifier la saillance relative. Obtenir cette information de saillance relative à partir de la méthode précédente n'est en effet pas toujours possible. Cela dépend de la configuration respective des trois entités de la scène précédemment évoquées. Pour résoudre ce problème, nous élaborons une méthode dont le cœur est un réseau de neurones récurrent faiblement supervisé. Le rôle de ce réseau est de représenter les trajectoires avec un code latent de faible dimension, facilitant la décision ultérieure. La faible supervision est réalisée avec une structure d'auto-encodeur, complétée par une contrainte de cohérence ajoutée à la fonction de perte. Cette contrainte traduit le fait que les trajectoires normales sont semblables, et sa fonction est de rapprocher leurs codes dans l'espace de représentation. La classification des trajectoires en saillantes ou non saillantes s'appuie sur une comparaison des codes au code médian dans l'espace de représentation. Des expériences menées sur des données synthétiques et réelles ont permis d'évaluer les

performances de la méthode. Les trajectoires réelles sont des trajectoires de piétons extraites dans la gare de Lausanne avec un réseau de caméras et un algorithme de suivi dédié présentés dans [ARF14]. En particulier, inclure la contrainte de cohérence permet dans la majorité des cas d'obtenir de meilleures performances. Cet effet est plus visible lorsque la différence entre la saillance et la normalité est plus prononcée.

Annexe A

Dans l'annexe A, nous présentons des expériences complémentaires concernant l'estimation de la saillance de mouvement relative directement depuis des vidéos. La saillance relative émerge lorsqu'un flot principal d'objets mobiles est présent et que quelques objets mobiles ne suivent pas ce flot. Il n'existe à notre connaissance pas de base de données de vidéos réelles dédiées à ce problème. Nous avons donc construit une base de vidéos 3D synthétiques pour des fins d'apprentissage et d'évaluation. Nous étudions deux approches pour cette tâche. La première repose sur l'exploitation du flot optique. La seconde s'appuie sur une segmentation de la scène en objets, puis sur le suivi de ces objets pour en extraire les trajectoires.

Perspectives

Plusieurs idées peuvent être explorées pour améliorer et étendre les travaux présentés dans cette thèse. Les perspectives de court terme incluent le test de nouvelles architectures pour les différentes méthodes. C'est bien sûr le cas des différents réseaux convolutifs utilisés, qui pourraient bénéficier d'une exploration plus poussée de l'espace des configurations. L'amélioration des autres composantes pourrait également conduire à de meilleures performances. C'est par exemple le cas de l'estimation du mouvement dominant pour la détection de la saillance, actuellement réalisée avec un modèle affine. Les perspectives de plus long terme incluent la capacité à séparer la saillance de profondeur de la saillance du mouvement pour l'estimation de cartes de saillance. S'appuyer sur de l'apprentissage est une piste envisageable pour résoudre ce problème. Enfin, il serait intéressant d'explorer des mécanismes d'apprentissage faiblement supervisé comme celui proposé pour la saillance de trajectoires à d'autres problèmes, comme par exemple la détection d'anomalies.

CHAPTER 1

INTRODUCTION

1.1 Context and motivations

In order to guide our actions in the environment in which we evolve, we dispose of our senses, classically classified into vision, hearing, touch, taste and smell, and to which we can add several other ones such as for instance the sense of equilibrium or the ability to feel the temperature. Some animal species even dispose of additional senses, such as magnetoreception for some cetaceans or birds, which allows them to perceive magnetic fields, or echolocation for bats or dolphin notably, which allows them to use sound to probe their surroundings. Various and complex sensors are the foundation of these senses. Among all these senses, vision is of particular interest. Indeed, in an environment filled with air, void or even shallow water, it provides rich and detailed information about surrounding objects. Moreover, by relying on the perception of light, vision can provide information about distant objects almost instantly. For this reason, we tend to largely rely on this sense to analyse and navigate in our environment.

Even when focusing on vision alone, the environment is extremely complex. It is common to see simultaneously many objects or people, all of them potentially requiring an action from the observer. Birds should rather flee if they see a cat, and a sheep may be tempted by rich grass in a pasture. Pedestrians should be careful when crossing a road, and cars should avoid collisions with static or moving objects. A first solution to properly handle such complex environments would be to permanently pay attention to each and every element of the whole scene. While this would solve the problem at hand, this solution is not the one we apply in practice. First of all, our field of view is limited. In this field of view, our central vision, that relies on the region of the eye called the fovea, is sharper than our peripheral vision. This alone means that we must move our head and eyes if we want to analyse fully our environment, and in particular if we want to focus on one element. Moreover, a permanent full analysis of the viewed scene would require a high energy cost, without mentioning the need of a higher complexity of the brain to conduct the analysis.

To permanently analyse the environment at a reasonable cost, we rely on visual attention. Visual attention is triggered by both the external environment and the brain. Its role is to select important information by relying on saliency cues. Once relevant elements are identified, we can focus on them and dedicate our resource to analyse them.

A first family of saliency cues is linked to the appearance of objects. For instance, a few red raspberries will be salient among the green bushes, green trees and green grass. In a scene with a majority of similarly looking objects, we are likely to be already aware of the presence of these objects, and to have undertaken the required actions if necessary. There is no particular reason to look more specifically at one of these objects than at another one. On the other hand, if there is an additional object with a very different appearance, the only way to analyse this specific object is to focus on it. It is then possible to decide whether this object requires attention or if we can ignore it. Saliency is in this case simply defined by a comparison with a normal appearance in the scene.

FIGURE 1.1 – Saliency determined with appearance cues (first line), and saliency determined with motion cues (second line). The horse and the sign are clearly distinct from their surrounding, which is by design for the sign. The two chameleons in the bottom left image are far easier to distinguish when they are seen in motion than in a static image such as this one. If a pedestrian were crossing the road in the bottom right setting, the cars motion would be a key element to pay attention to. The images are from the MSRA-B [Wan+17], Camouflaged Animals [BLM16] and FBMS-59 [OMB14] datasets.

A second family of saliency cues is linked to motion. First, by looking at a static scene, it can take only a few instants to overview the scene, and to analyse or decide to ignore its content. Then, if nothing moves in the scene, it means that there are no new elements, and that the past analysis still holds. It is then unnecessary to spend more energy to analyse again any part of the scene. On the contrary, if a moving object enters the field of view, this object has not been analysed and deserves more attention than the rest of the scene. More generally,

any moving object of the environment presents an interest. Because of its motion, it may interact with the viewer, either by coming to him/her, fleeing him/her, hunting him/her, or simply potentially crossing his/her path. In all these cases, the viewer should undertake the appropriate action. Leveraging motion indices is then a relevant way to select the element on which to focus. If there are several moving objects, motion saliency cues should provide the most significant highlight on the object with the most distinctive motion, which can be for instance the fastest/slowest one, the one with a distinct motion pattern, or the one following an erratic trajectory, and which is the most likely to require a specific response. Motion cues are based on the evolution of the scene across time. Naturally, if the head or eyes of the viewer move, this will induce an apparent motion for the static objects as well. Nevertheless, this phenomenon is mitigated in human and animal perception. However, it must be explicitly taken into account in artificial vision systems.

Appearance and motion yield two useful but different kinds of saliency cues, further illustrated in Figure 1.1. Appearance cues drive attention toward elements with potentially specific shapes or colors. Motion cues trigger attention toward moving elements undergoing specific motion or action.

The discussion above was focused on the specific role of vision for the perception of the environment of people and many animals, and on the importance of saliency to allow for an efficient processing of the input signal. Now, if we consider the problem of building robotics systems able to act autonomously in a given environment, they should be able to conduct a similar analysis of their surroundings. Such systems are commonly built with constrained resources, namely limited energy consumption, processing time and processor load, thus making an efficient processing needed. We can also consider the broader problem of video analysis, which includes tasks such as video surveillance, video summary or crowd motion analysis. In all these cases, the detailed analysis should be restricted to the most interesting parts to avoid wasting computing resources. The particular relevance of motion cues for all these tasks is due to the fact that it provides information about the dynamics of objects in the scene. It it then the adequate piece of information to either ensure a safe navigation, or to monitor and detect unusual and potentially dangerous behaviours. The objective of this thesis is precisely to consider and analyse the problem of motion saliency estimation in video sequences.

1.2 Related work

Related work will be discussed in details in a dedicated chapter. Here, let us simply present an overview of the most notable points.

Saliency estimation is a frequent concern. It may operate as a pre-attention mechanism or be a goal in itself. When working with image or video data, its output usually consists of saliency

maps providing the probability of saliency at each pixel. Actually, saliency estimation is close to visual attention, object detection, or even image segmentation. There is not really a single formal definition of image saliency, which can have several declinations. We already made the distinction between appearance and motion saliency. In the literature, the following declinations of saliency are commonly used. A first possibility is to consider that what attracts the attention of human observers is salient. This can be recorded with specific eye-tracking devices, which measure the spatial location where people look at when they view videos. A second possibility is to consider that objects with very distinctive appearances and motion are salient. Typically, objects located in the foreground and with a pronounced motion are considered to be salient.

For these definitions of saliency, both appearance and motion play a role for the identification of the elements of interest. It is then natural for saliency estimation methods targeted at these tasks to exploit both. Regarding appearance, information about colors, textures, border and shape are typically used. Regarding motion, the temporal consistency, the optical flow, from which the displacement magnitude or motion boundaries can be extracted, are commonly leveraged. The difference between the different methods mainly lies in the way these elements are extracted and combined to estimate saliency. This can be done by designing a dedicated framework to put these pieces together. Another possibility is to rely on deep neural networks, which have proven very effective to handle image data and which are adopted by most recent methods.

More generally, elements deviating from the normal or whose behaviour is unexpected can be considered to be salient. When dealing with this generic definition, it is more common to speak of anomaly detection. Anomaly detection extends to non image data as well. Trajectories are an example of such data, which is relevant for our problem as they convey motion information. For this family of problems, the goal will be to find anomalous patterns in the data, which can be done through the exploitation of domain knowledge, the use of statistical tools or again with deep neural networks.

1.3 Overall contributions

Although motion saliency cues present a clear interest for scene analysis, existing methods commonly mix motion saliency with appearance saliency to produce a unique combined saliency estimate. This is unfortunate, since it hides the cause of the saliency decision. For examples given above (navigation for mobile robotics, video surveillance...), we may be interested in motion saliency only. In this case, appearance saliency should not act as a distractor. In fact, there are situations where the appearance does not play any role. A typical example is an individual walking against a crowd. It can occur in many diverse applications, such as urban and road traffic monitoring [RB19 ; BFM19], ensuring safety of public areas prone to dense

crowd [PRBB17], studying cell dynamics in bio-imaging [Rou+17], or identifying adverse weather conditions in meteorological image sequences [Pap+00]. In all these cases, motion is likely the only element of concern to extract relevant information about the viewed scene.

When dealing with motion saliency, three kinds of entities are usually involved :

— The static scene,
— Salient moving elements,
— Normal moving elements.

Since we are interested in motion saliency, the static scene is obviously not likely to be salient. In contrast, objects undergoing a singular motion are salient. It is also possible to encounter situations where several or many objects are moving and follow the same motion pattern. In this case, they will be considered as normal. Salient moving elements will be objects undergoing motion departing from the main one. These different groups of objects may be perceived by a camera that can be either static or itself moving. In the later case, apparent motion will be induced over the whole image, modifying the configuration of motions in the video. Our work will address these issues in the following chapters.

The contributions of this thesis are organised around three axes of motion saliency analysis.

The first one is concerned with the temporal detection of motion saliency in video sequences. The goal is to be able to decide which frames contain no motion saliency and can then be further ignored, and which frames do contain motion saliency. To our knowledge, this problem has not been investigated so far. It is however a useful pre-requisite for saliency localisation methods, which often assume that saliency is present. Our purpose is then to build a pre-attention mechanism that can be used to trigger further processing of the video frames, exclusively when it is relevant.

The second axis deals with the estimation of motion saliency maps (see the illustration in Figure 1.2). When motion saliency is found present, it is useful to locate the salient element in the image. An autonomous mobile robotics system could then use the estimated map to decide whether and how to adapt its trajectory, and a video analysis system can highlight to a human operator the location of the element of interest.

The third axis focuses on the estimation of trajectory saliency. Long-lasting motion saliency can appear only after some delay, and trajectories are a good way to capture lasting events, as illustrated in Figure 1.3. The trajectories can be extracted from videos with tracking systems. Trajectories are however not restricted to this setting, as they can be provided by sensors such as GPS, nowadays commonly present in cars or mobile phones. More generally, any time series can be viewed as a trajectory and processed similarly. The goal of this third axis is to define a method to estimate trajectory saliency, that is, in a group consisted mostly of similar normal trajectories, finding the few dissimilar ones.

FIGURE 1.2 – Ground-truth motion saliency maps associated to four examples. A first possibility is that no element of the scene undergoes any salient motion. In this case, we get an empty ground truth saliency map. The two first samples correspond to such a situation. We acquired these videos as part of the dataset described in Chapter 3. In contrast, salient moving elements are present in the two last samples, as highlighted in the ground truth. The two last samples come from the DAVIS 2016 dataset [Per+16].

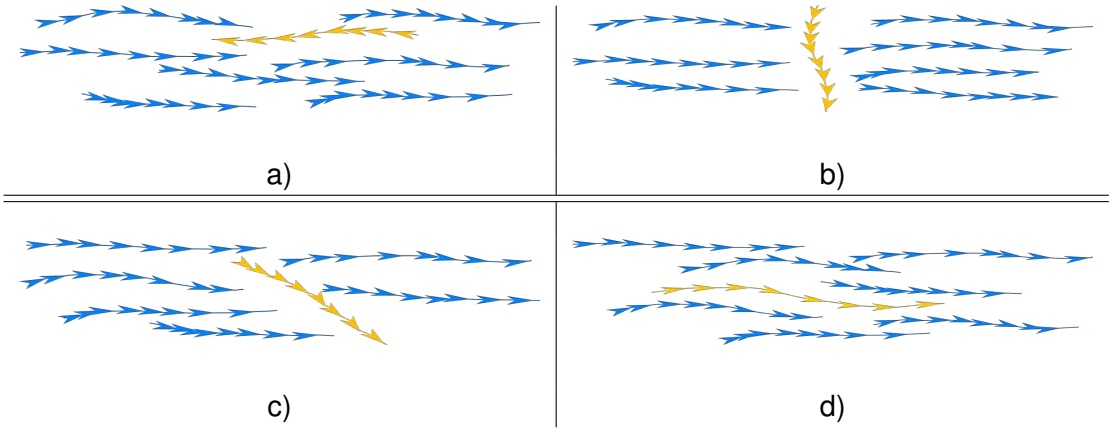


FIGURE 1.3 – Illustration of four trajectory saliency settings. Normal trajectories are drawn in blue and salient ones in orange. In settings a) and b), the salient trajectory is clearly distinct from normal ones. In setting c), the salient trajectory has a slightly different direction, and in setting d) the salient trajectory corresponds to a slightly faster motion compared to normality. For the two last settings, motion saliency is less pronounced. Looking at the motion at only one instant may not be sufficient to correctly discriminate salient ones, especially if motions are not regular enough. An observation covering a longer time period, as provided by trajectories, will help making better predictions.

1.4 Organisation of the manuscript

The manuscript is organised in four main chapters. Chapter 2 consists in a survey of the relevant literature. In Chapter 3, we develop a method for frame-based motion saliency detection. Chapter 4 introduces a method for motion saliency map estimation based on optical flow inpainting. Chapter 5 is focused on trajectory saliency. Additionally, Appendix A investigates relative motion saliency with a 3D video dataset. Below, we present a brief overview of each of these parts.

Chapter 2

In Chapter 2, we describe related work regarding saliency estimation and anomaly detection, which are fields closely related to motion saliency estimation. This survey allows us to identify research fields related to our overall objective of defining motion saliency estimation methods. Several fields addressed in this chapter will be explored to build our methods, that is, deep neural networks, optical flow estimation, deep metric learning and methods for learning with weak supervision.

Chapter 3

In Chapter 3, we investigate the problem of frame-based motion saliency detection, that is, deciding which frames of a video contain motion saliency. The method we propose involves two steps. First, the dominant motion due to the camera displacement is cancelled. Then, the classification into the salient or non salient class is achieved with a convolutional neural network. We experiment working with either raw images or the optical flow in our workflow. Additionally, we consider two methods for the estimation of the dominant motion in the image due to the camera motion. This gives a total of four variants, that will be compared and evaluated. The training and the experimental evaluation of our method variants rely on a synthetic dataset of image pairs with known motion and a dataset of real videos. The experimental results will show that we are able to correctly detect motion saliency, even in presence of static elements which may have a strong apparent motion due to the camera motion and their location in the foreground of the scene.

Chapter 4

In Chapter 4, we address the problem of motion saliency map estimation, which consists in predicting a saliency value for each pixel of the video frames. To solve it, we will elaborate a method relying on optical flow inpainting. In a first step, candidate salient regions are extracted. Then, the optical flow inside these regions is inpainted from the surrounding flow. The idea is

that, if the region is salient, the inpainted flow should be clearly distinct from the flow computed originally in that region. In contrast, if the region is not salient, the reconstructed flow and the original one should be similar. We leverage the difference between these two flows to compute the motion saliency map. Experimental results will demonstrate that our method compares favourably against existing methods.

Chapter 5

In Chapter 5, we tackle the problem of trajectory saliency estimation. Trajectories enable to handle motion saliency that progressively appears over time. Moreover, when several moving objects are present, trajectories allow us to compare more easily the respective motions, and consequently to find relative motion saliency. It would be less immediate from raw optical flows. In addition, it may be not accurate enough to lead to a precise identification of all the moving objects in the image. We will develop a framework whose core is a weakly supervised recurrent neural network. The role of this network is to represent trajectories with a latent code, so that similar trajectories should be represented with close codes, and dissimilar trajectories with distant codes. The weak supervision is achieved with an auto-encoder structure, complemented by a consistency constraint added to the loss. This constraint expresses that non salient trajectories are similar, and its role is to make non salient codes closer in the embedding space. Experiments will be carried out on synthetic and real datasets.

Appendix A

In Appendix A, we present further (preliminary) investigations regarding the estimation of relative motion saliency from video sequences. Relative motion saliency occurs when there is a main stream of moving objects, and a few salient objects exhibiting different motions. To our knowledge, there is no real video dataset with ground truth dedicated to this problem. Then, we have built a 3D synthetic video dataset for training and evaluation purpose. We have designed two methods for this task. The first one is based on optical flow. The second one requires the segmentation of the scene into objects. The successive segmentation throughout the video will supply their trajectories. The second method will yield encouraging results, but also raises still open issues.

CHAPTER 2

MOTION SALIENCY : RELATED WORK

In this chapter, we review work related to motion saliency estimation. We will start by considering two research fields on which many image and video processing methods rely, and that will be particularly relevant for motion saliency estimation. The first of these fields is deep learning. Section 2.1 will summarise its most notable theoretical and practical aspects. We further discuss two aspects of deep learning in Sections 2.2 and 2.3. Section 2.2 will be dedicated to work concerning weak supervision for deep learning, and Section 2.3 will address the field of deep metric learning.

The second of the research fields commonly leveraged when estimating motion saliency is the estimation of motion in videos through optical flow. It will be discussed in Section 2.4.

The review will then focus on several declinations of saliency estimation in Section 2.5, namely, eye-tracking saliency, static image saliency, video saliency and anomaly detection. Most of these tasks are related to motion, and they all have in common the goal of identifying elements departing from their context.

We will finally discuss in Section 2.6 work related to trajectory analysis. Indeed, trajectories represent a natural approach to handle long-term information, and then to estimate long-term motion saliency. Section 2.7 contains concluding comments.

2.1 Deep neural networks

Artificial neural networks have been developed with a double objective. The first one is to propose a model of natural neural networks found in the brain. The second one is to exploit this model to make it possible for machines to reproduce some of the brain functionalities. We will focus on what has been achieved for this latter objective. In Section 2.1.1, important milestones as well as core elements defining artificial neural networks will be discussed. Section 2.1.2 will be centred on additional considerations for the successful usage of neural networks in practice.

2.1.1 Brief history and core elements of deep neural networks

The perceptron, proposed by [Ros58] in 1958, is an early example of artificial neural network. A perceptron is composed of neurons. These neurons process the input through linear

combinations and non-linear thresholding operations to produce the output value. The perceptron was developed to serve as a (simple) model of the brain.

A major milestone in the history of neural networks is a theoretical result regarding their modelling capabilities. In [HSW+89], the authors proved mathematically that, if a neural network contains at least one hidden layer, uses an arbitrary squashing function, and has a sufficiently large number of hidden units, it is able to approximate any function from one finite dimensional space to another which is Borel measurable. The functions considered in practice are expected to be Borel measurable, which means that neural networks can be considered as universal function approximators.

Following these theoretical guarantees, the next question was how to successfully apply neural networks to practical problems.

A key element is the architecture of the neural network. Let us discuss convolutional neural network, that were originally designed for image classification and that are a central element in the success of deep networks for image processing applications. For the image classification task, the input is an image, consisting in a grid of pixels whose colors are represented by a (Red, Green, Blue) triplet. For each class we want to recognise, for instance digits, cats, people, etc., the network typically produces a score. The prediction of the network will be the class associated to the highest score. The question is then how to map the input, which is a 2-dimensional image, to the output, a 1-dimensional vector with as many components as the number of classes. The core ingredient is a succession of 2D convolutions, each of them being followed by a non linearity. The role of the 2D convolutions is to leverage the spatial structure of the image. The non linearity, which can be for instance a sigmoid, an hyperbolic tangent or simply a ReLU, is required to make it possible for the network to approximate any mapping, and not only linear ones. The successive layers learn more and more abstract features, while reducing the size of the feature map. The final score vector is then obtained with fully connected layers. The famous LeNet network [LeC+98], represented in Figure 2.1, illustrates these general design principles.

Over the years, it has been found that neural networks with a larger number of layers can lead to better performance. These neural networks were then qualified as “deep”. Note that depending on the context, a network with as few as three layers may be qualified as deep. By extension, the expression “deep learning” is commonly used to qualify methods that rely on the training of deep neural networks.

Architectures following the above-mentioned principles necessarily comprise a huge number of parameters, which can easily reach hundreds of thousands or even several millions. At this point, the question is how to set values for all these parameters that would produce a relevant mapping between input and output. Training is required which is often based on Stochastic Gradient Descent (SGD). The idea is to define a loss function to minimise by an iterative pro-

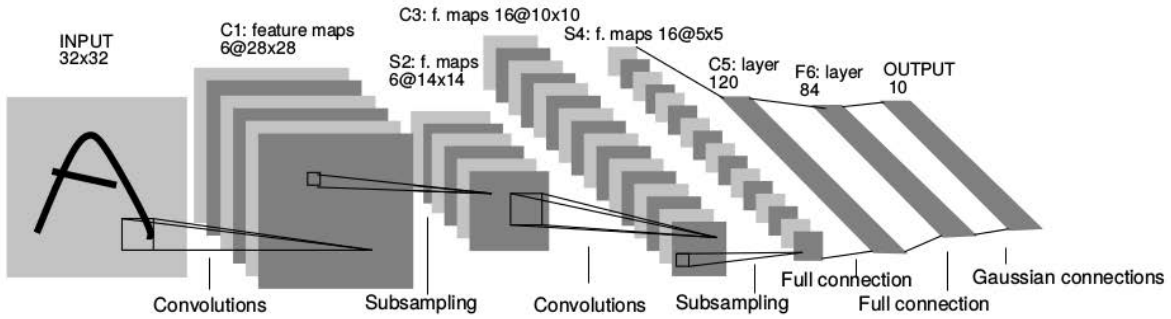


FIGURE 2.1 – Architecture of the LeNet-5 convolutional neural network designed for digit recognition. Reproduced from [LeC+98].

cess. A few samples are grouped into a batch, and the weights are updated to minimise the loss for this batch with a gradient descent algorithm. This is repeated until some convergence is reached.

An example of loss frequently used for image classification is the cross-entropy loss. Let us assume that we have a network taking images as input and that produces an output vector of dimension C, with C the number of classes considered. The output vector stores a score x_i for each of the C classes. These scores $x_i \in \mathbb{R}$ can be converted into estimated probabilities $p_i \in [0, 1]$ that the image corresponds to the i^{th} class by taking :

$$p_i = \frac{\exp(x_i)}{\sum_{j=1}^{C} \exp(x_j)}. \tag{2.1}$$

In standard classification setups, the classes are often considered to be mutually exclusive, this is why the probabilities are defined so that their sum is equal to 1. The cross-entropy loss for a sample image whose class index is c is then defined by :

$$\begin{aligned} \mathcal{L} &= -\log(p_c) \\ &= -\log\left(\frac{\exp(x_c)}{\sum_{j=1}^{C} \exp(x_j)}\right) \\ &= -x_c + \log\left(\sum_{j=1}^{C} \exp(x_j)\right) \end{aligned} \tag{2.2}$$

A validation set can help monitoring the training stage. The most frequently used optimisation methods include Adagrad [DHS11], RMS Prop [TH12] or Adam [KB14].

The labelled training data is an important part of the optimisation process. Indeed, the availability of a huge training dataset has proven to be crucial for the success of deep learning methods. Examples of datasets include MNIST [LeC+98], Cifar [KH+09], or ImageNet [Rus+15]

for image classification, Kitti [GLU12], Flying Chairs [Dos+15] or Sintel [But+12] for optical flow estimation, FBMS [OMB14] and DAVIS [Per+16] for video object segmentation. In particular, the impressive breakthrough obtained by [KSH12] for image classification was made possible by leveraging the very large ImageNet dataset, that includes millions of labelled images. In [Sun+17], the authors found that performance increases logarithmically based on volume of training data. To reach this conclusion, they considered a dataset of 300 million images with noisy labels. They even found that for a large dataset generated automatically but involving noise, the negative impact of the noise is more than counterbalanced by the scale of the data. Still, it is in practice not always easy to get a large annotated dataset, and this point remains an open challenge that we will further discuss in Section 2.2.

A consequence of the requirement of a huge dataset is that a fair amount of computational power is required to train deep networks. In [KSH12], the authors were able to leverage the ImageNet dataset with millions of images by optimising their neural network on a GPU (graphical processing unit). GPUs are able to handle efficiently many operations in parallel, which is very beneficial for deep neural networks designed for image processing. Since then, many framework have been developed to allow for a quick and efficient design of deep networks, such as Theano [Ber+11], Caffe [Jia+14], Keras [Cho15], Tensorflow [Aba+16] and PyTorch [Pas+19]. Thanks to the efforts that have been made to optimise both the hardware and software, typical deep network architectures can now run in real time.

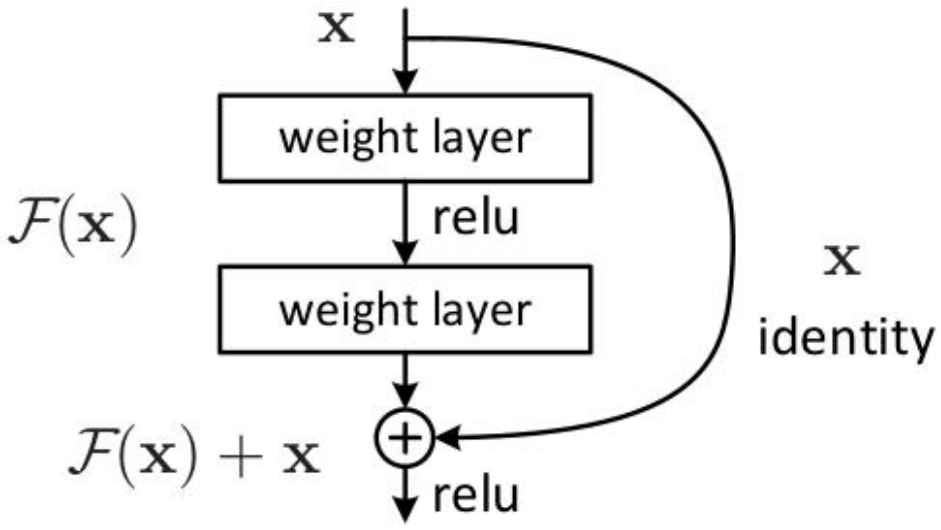


FIGURE 2.2 – Architecture of a residual block. The output of the block is the output of its second convolutional layer, added to the input of its first convolutional layer. This structure allows the gradient to flow easily through the whole residual network, which is composed of a succession of residual blocks. Reproduced from [He+16].

Since the work described in [KSH12], deep architectures were improved for the image clas-

sification task, notably in [Sze+15 ; He+16 ; Hua+17]. There has been a trend to develop deeper and deeper nets, which can even reach several hundred layers [He+16]. A notable trick consists in adding shortcuts between layers to ensure that the gradient-based optimisation can update all the weights. This takes the form of residual connection in [He+16] (see Figure 2.2) and of dense connections in [Hua+17].

After their successful application to image classification, deep networks have been investigated for other computer vision tasks. The success they encountered often led to a paradigm shift in these fields. It is then common to make the distinction between deep-learning based methods and classical methods.

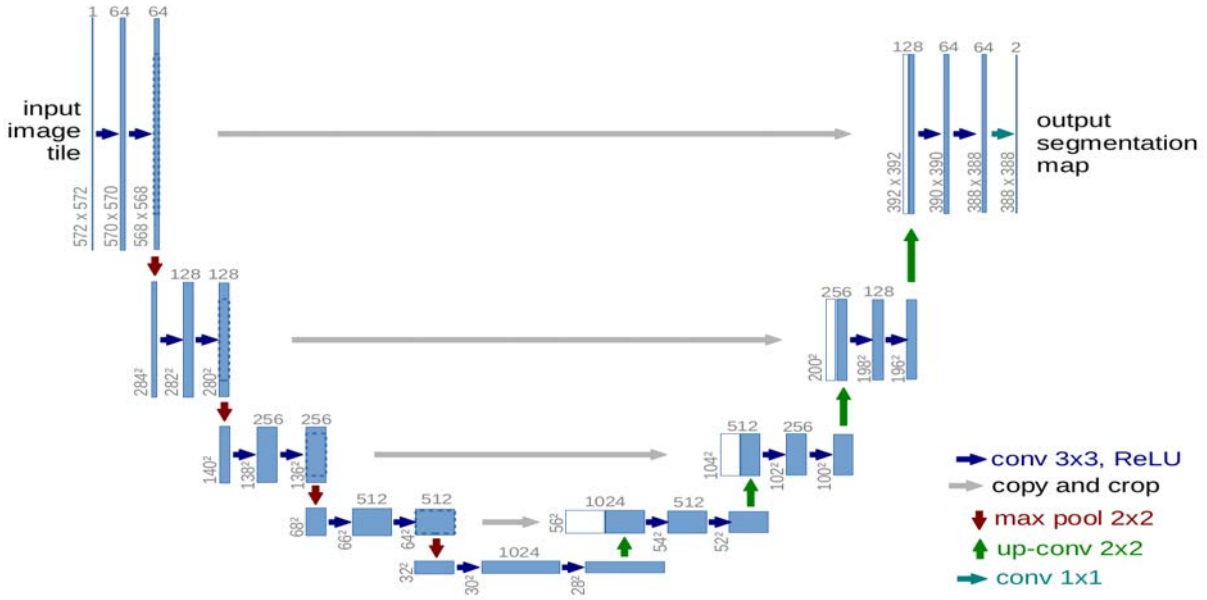


FIGURE 2.3 – Architecture of the U-Net network for biomedical image segmentation. The network is constituted of a contracting part, whose role is to learn features at a limited computational cost, and an expanding part, which ensures that the prediction has the same size as the original image. Reproduced from [RFB15].

To process sequential data, a family of networks has been developed, namely recurrent neural networks. These networks include memory cells, whose role is to hold and update past information. Recurrent neural networks such as the LSTM, proposed originally in [HS97], have been designed for the processing of sequential data. Such networks were applied to applications such as speech recognition [FGS07], natural language processing [Kum+16], but also on video processing [JXW17 ; Lai+20].

Let us mention a few examples of other applications to conclude this section. In [RFB15], the authors applied deep networks to biomedical image segmentation. A central architectural change consists in relying on operations that allow to predict an output map of the same size

as the input one instead of an output vector. For this, the authors rely on an upsampling of the feature map followed by standard convolutional layers (see Figure 2.3). In [Che+18b], the authors rely on a similar idea, by proposing to use dilated convolutions to upsample the feature map. The work of [He+17] is an example of application to the task of object instance segmentation, with a deep network which, in addition to recognizing the object classes present in the image, separates the different objects belonging to the same class. Deep networks have also been successfully applied to video processing tasks, as for instance automated description of videos [Yao+15], or action recognition in videos [SZ14 ; SKS16 ; Li+18].

2.1.2 Performance and computational cost issues

The design and training of deep neural networks remain in part an empirical process. The application of appropriate tricks can in practice lead to improved performance. These tricks can be used when defining and testing new architectures, but some of them can as well be used to improve existing methods. This section discusses a few notable tricks.

In [Ben12], the author makes practical recommendations for the training of a deep network. Many options or hyper-parameters are involved : the pre-processing of the input data, the choice of the learning algorithm, in turn requiring to set parameters such as the learning rate, the batch size or the training schedule. Even without taking into account the multiple possible architectures for the network, there are already many choices to fix. The author notes that all these hyperparameters do not have the same impact on performance. For instance, properly setting the learning rate is often critical. On the other hand, the batch size is mainly linked to computational efficiency (it is generally desirable to feed simultaneously to the GPU as many images as possible). The search of the right hyperparameters can be done with a grid search. However, the high number of possible combinations of hyperparameters makes an exhaustive investigation with this approach not tractable. Another option proposed by [Ben12] is to perform a random search in the hyperparameter space. The author found it better than exploring the many possible configurations. In [Jad+17], the authors went further in this direction by defining the Population-Based Training algorithm. This algorithm leverages large computing resources to train several models in parallel. The idea is to train a population of models. Regularly, the lowest-performing models are discarded. They are replaced by the best-performing models whose weights and hyperparameters are slightly modified before resuming training. This approach provided better results and faster convergence than grid and random search.

To further emphasize the impact of training, let us mention the following point. Published results tend to be considered as the best possible results obtainable for the models described, but this is not necessarily the case. In the context of optical flow estimation, the authors of [Sun+20] provide an empirical analysis of the impact of training. By modifying the training procedure (notably by including learning rate disruption), they could reach a higher precision with

a FlowNetC network than with the original FlowNet 2.0 network [Ilg+17], while FlowNetC is a subnet of FlowNet 2.0.

A deep network should not only deliver accurate results, it should also have a limited memory footprint, which is particularly critical for embedded systems with limited resources such as mobile phones. Large neural networks can be pruned to 10-20% of the number of their original weights, while maintaining the same level of performance [FC18]. However, retraining the pruned model from scratch generally yields lower performance. To explain this, the authors formulate the lottery ticket hypothesis, according to which the lucky initialisation of a sub-network is what enables good performance. Such a lucky initialisation is more likely to happen in a larger network, which comprises many sub-networks.

The loss function plays an important role for obtaining good performance. It ensures that the network gains relevant information from the data through back-propagation. This is particularly true for tasks requiring a prediction for all pixels of images. For such tasks, classical losses such as the cross-entropy loss, which is computed and summed for all the pixels of the image, may provide a comparatively negligible learning signal for small misclassified elements. In [Lin+17], the authors propose the focal loss to address this issue. This loss is prone to increase the value of the gradient for misclassified pixels compared to the value of the gradient for properly classified pixels. By doing so, even small objects can have a significant contribution to the model update, which leads to performance improvement. Another important concern regarding the loss function is its adequacy with the final objective. Generally, one or several metrics are used to evaluate how well the network performs. In [Bru+20], the authors consider the problem of visual saliency estimation. They find that a loss consisting of a linear combination of several terms, individually focusing on a different aspect, with for instance a pixel-based term, a distribution-based term or a saliency-inspired term, allows to improve the performances. In particular, performances are more evenly distributed over all the metrics, that emphasize each a different point.

2.2 Weak supervision for deep learning

As we have seen so far, deep learning methods are successfully used for a wide range of tasks. However, a major constraint is the need of large training datasets, which usually require expensive annotation efforts to be built. Research works have then been dedicated to reduce the amount of supervision needed to train deep neural networks. With weak supervision, it is easier to make a method applicable to new configurations, or to increase the amount of training data and improve performance. Weakly supervised networks are also relevant for anomaly detection in general, and for motion saliency in particular.

Let us first highlight an important point raised in [Rol+17]. In a supervised setting, deep

neural networks can be trained successfully even with a high proportion of erroneous training labels. For the unsupervised setting, it means that the learning signal, which depends on the choice of the loss function applied to the data, may also be noisy. In [Pap+15], the authors take advantage of this property. They develop a method for semantic image segmentation with low supervision. It involves an inaccurate annotation (bounding box or even image-level label), combined with a few strongly labelled images to further improve performance. A probabilistic approach is used to define the loss from the coarse labels. Another kind of abundant noisy data is leveraged in [Mah+18] for pre-training. The authors designed a network to predict the hashtags associated to billions of images extracted from social media. This kind of data can be obtained with weak annotation cost.

Another way to get information to train a network in an unsupervised way is to find a relevant pretext task, which is a task not directly related to the final objective, but that requires no supervision for training. In [GSK18], the authors randomly rotate images by an angle $\theta \in [0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$, and train a network to predict the angle. The idea is that, to be able to solve this problem, the network should use the canonical orientation of the objects present in the image. This means that the network should be able to extract relevant features to characterise objects. Following a similar idea, the authors of [NF16] construct jigsaw puzzles from images. They build a network to predict the position of the image patches composing the jigsaw puzzle. The network is expected to learn a representation of the spatial organisation of images, which can be reused for other tasks.

Weak supervision can also be achieved by leveraging multi-modal data. In [Nav+19], the authors develop a robotic system equipped with a camera and a short-range sensor. The authors make use of the short-range proximity sensor to label training data for the camera. In [Par+19], the authors combine audio and visual information with a weakly supervised scheme. For this, they leverage video-level label with no precise spatial or temporal information. In [Sun+19], the authors defined the videoBERT method that learns from visual and text input. Their objective is to learn a joint representation. The supervision is obtained by masking a part of the signal and by trying to predict it (for instance, masking a word in a sentence). They find that leveraging multi-modal information helps to improve performance.

The authors in [Wan+19a] achieve tracking within an unsupervised framework. They rely on the fact that tracking an object forward and then backward should give the initial position of the object.

Finally, another family of unsupervised methods is based on auto-encoder networks, and more specifically Generative Adversarial Networks (GANs) [Goo+14]. Two networks are trained together by competing against each other. The first network is a generator network, whose goal is to generate samples not distinguishable from real samples. The second network is a discriminator network, which tries to predict whether its input samples have been produced by

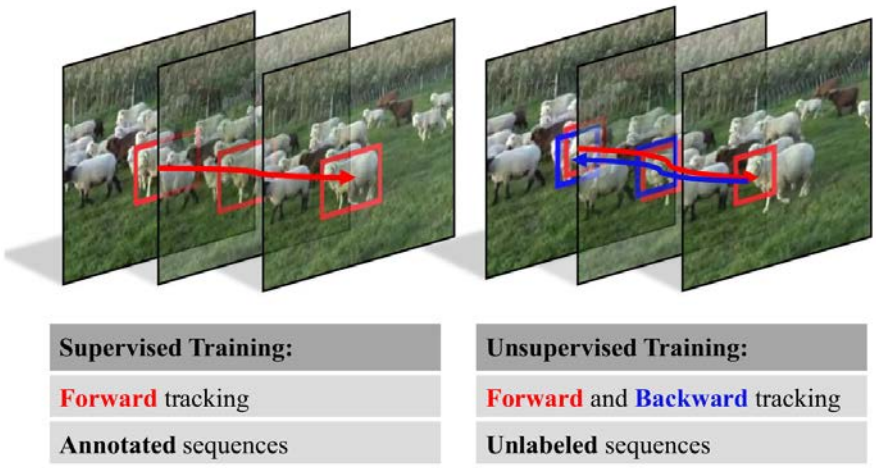


FIGURE 2.4 – Trick for unsupervised learning for tracking proposed in [Wan+19a]. The authors track the same element forward and backward. This process should lead to the initial position of the tracked element, which is used to define the unsupervised training algorithm. Reproduced from [Wan+19a].

the generator or whether they are real samples. The training of GANs is known to be tricky, and methods have been proposed for a better convergence, such as in [Sal+16]. In the context of the trajectory prediction task, the GAN framework has been used as a means to train a model without requiring manual annotations [AHP19]. In this work, the authors make use of an InfoGAN [Che+16] to avoid the problem of mode collapsing, which may otherwise be encountered with GAN networks.

The reader is referred to the survey [JT20] for a deeper analysis.

2.3 Deep metric learning

Deep neural networks are able to approximate practically any function from a finite dimensional input space to a finite dimensional output space [HSW+89]. As discussed in Section 2.1, for image classification, the output consists in general of a vector with as many components as the number of classes. The value of each component represents the probability that the sample corresponds to a particular class. However, this way of proceeding is only one of the many possibilities that could be considered. It also has drawbacks : to be able to recognise a new class, the architecture of the network has to be modified and the network itself must at least be fine-tuned.

Instead of predicting an array of probabilities, another possibility is to predict a feature vector to embed each input sample. The task at hand will then be solved by processing this embedding. A common desirable property for the embedding is the following : similar samples should

be represented with similar embeddings, and dissimilar samples should be represented with dissimilar embeddings. Deep metric learning methods are methods that attempt to learn embeddings with such properties. For instance in [SKP15], the authors consider face recognition and clustering. For this, they propose to learn an embedding c of faces with a triplet loss. This loss, schematically represented in Figure 2.5, is defined from three samples. Two of them are pictures of the same person. They are called the anchor and positive sample, and they are represented with embeddings c_a and c_p respectively. The last sample is a picture of another person. It is the negative sample, represented with the embedding c_n. The triplet loss is then defined as :

$$L = \max(||c_a - c_p||_2^2 - ||c_a - c_n||_2^2 + \alpha, \quad 0). \tag{2.3}$$

The objective of this loss is to ensure the distance between c_a and c_p is smaller than the distance between c_a and c_n by a margin α. In [HA15], the authors develop a similar idea. They use a triplet network to learn such an embedding from the data. A triplet architecture is an architecture with three branches sharing the same weights, used to predict an embedding for three samples (the reference, the positive the negative one). They use the predicted embeddings to decide whether images represent the same class of object or not.

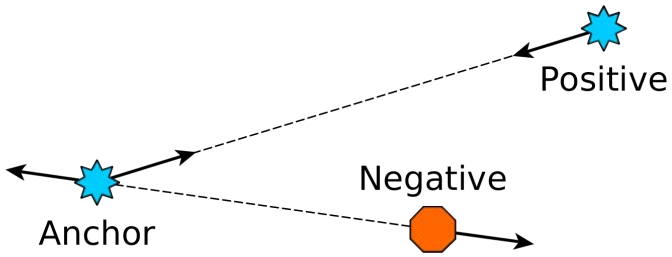


FIGURE 2.5 – Illustration of the triplet loss. Three samples are represented in the embedding space. The anchor and the positive samples are similar and should be represented with similar embeddings. The negative sample is different from the anchor, and the embeddings of the anchor and the negative element should be clearly distinct. The triplet loss acts as a force in the embedding space designed to enforce these properties.

In [HBL17], the authors warn that, for deep metric learning, the stagnation of the loss function does not mean the training should be stopped. Indeed, if the training continues for a while, it is quite common to see the network improving again. This behaviour is due to the fact that, while the codes are evolving in the embedding space, but are not yet positioned in clearly separated clusters representing the different classes, the classification step will not be able to reach good performance. The authors also note that to help the network learn, it is often necessary to mine hard triplets. Hard triplets are triplets for which making an accurate prediction is more difficult, for instance due to their similar appearance. They design a loss function that includes this

hard mining automatically to enforce this point. In fact, deep metric learning allows for a variety of loss functions to shape the embedding space. In [Wan+17], the authors consider the triangle formed by a triplet on the embedding space. They define an angular loss from this triangle, in order to achieve scale invariance. In [Wan+19c], the authors consider a list of (more than three) samples to build the loss. It allows them to extract more information about the structure of the data.

Let us now review how deep metric learning can be used in practice to solve a variety of tasks. A first application is the efficient processing of an image database. In [Wan+14], the authors rely on deep metric learning to estimate image similarity with the help of a learned embedding. Their goal is to retrieve images in the database images that are similar to the query. Moreover, this can be done efficiently by computing only once a compact embedding for each image. This embedding contains all the information required and can be processed at a much lower cost than the whole image. Another application is tracking. Given the initial location of an object of interest, the goal is to predict its location throughout the video. In [HLT16], the authors use deep metric learning to compute a similarity between samples in order to build their tracker. In [RT18], the authors consider a multi-target multi-camera tracking scenario. One notable challenge of this setting is the need to re-identify tracked people among many distractors. To solve it, they integrate in their framework deep metric learning, by leveraging a new adaptative weighted triplet loss and hard identity mining. In [ZDK19], the authors consider the problem of end-to-end place recognition. Their deep metric learning framework that includes an adaptative similarity metric for training, allows them to learn relevant features robust to appearance change.

The dimension of the embedding can *a priori* be chosen arbitrarily. In [NHD17], the authors adopt a 1-dimensional embedding to solve multi-person pose estimation and semantic segmentation. For pose estimation, the 1-dimensional embedding is used to separate the persons in the scene. The network learns to assign a unique number to each person, which proved adapted to the task. However, leveraging a higher-dimensional embedding can be more advantageous for more complex tasks. Each dimension can be interpreted as a feature, and it becomes unnecessary to predict increasingly large scalars to represent large numbers of elements. For semantic instance segmentation, [Fat+17] built a multi-dimensional embedding to separate object instances.

Similar approaches have been developed for the task of video object segmentation (VOS). In [Li+18b], the authors deal with an unsupervised setting, in which no indication of the object to segment is given. They use objectness score and optical flow to identify the regions of interest and to define seed points. Then, the embeddings are learned for instance segmentation on static images to estimate the similarity between pixels and to segment objects. The tracking works properly because the embedding, although being learned on static images, is stable

over time. In [Che+18c] and [Voi+19], the authors consider the semi-supervised setting of VOS, in which the ground truth masks of the objects to track are provided in the first frame of the video. The authors obtain state-of-the-art results, with very low computation time compared to competing approaches.

We refer the interested reader to [KB19] for a survey of the field of deep metric learning.

2.4 Optical flow estimation

The estimation of motion in videos provides valuable information for various computer vision tasks, such as moving object segmentation, object tracking or motion saliency estimation. This section will then discuss optical flow estimation. By definition, the optical flow provides for a given frame the apparent velocity at every pixel. The optical flow is a 2-dimensional field, given by the horizontal and the vertical component of the velocity vectors. It can also be visualised as a color image using a specific color code. Examples are drawn in Figure 2.6.

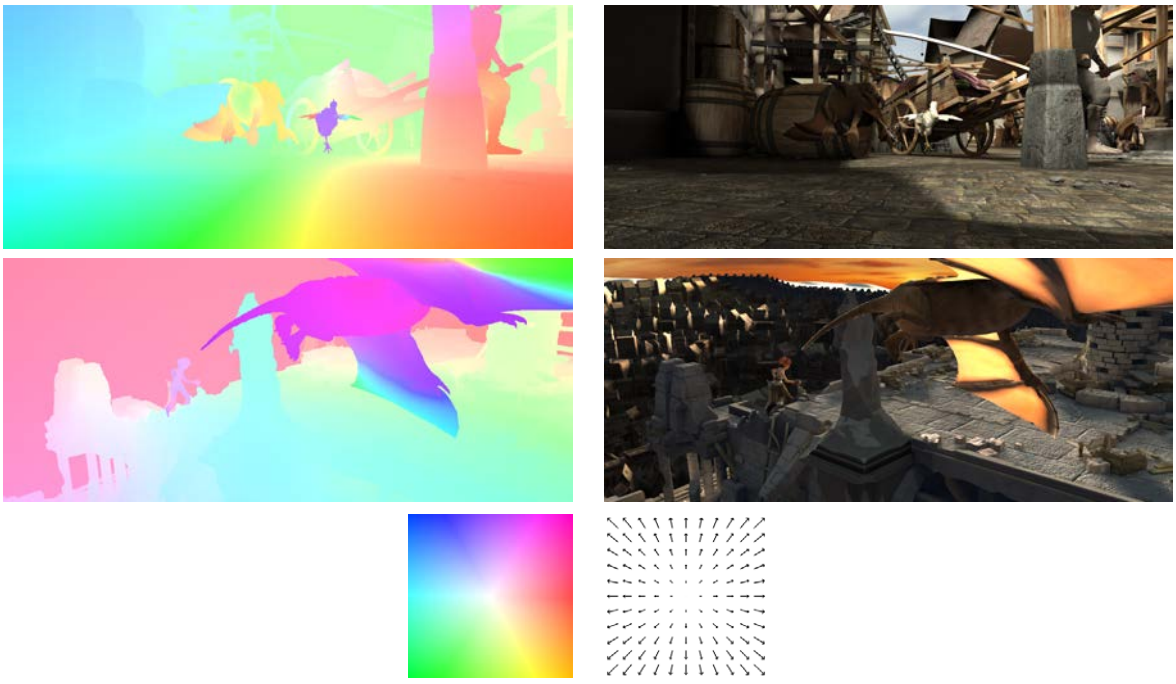


FIGURE 2.6 – The first two lines display the visualisation of the ground truth optical flow (left) and the corresponding frame (right). These samples are extracted from the MPI Sintel dataset [But+12]. The optical flow is represented with the HSV color representation, provided in the third line. The hue represents the motion direction, and the saturation represents the motion magnitude.

The pioneering work for optical flow estimation based on global optimisation was introduced

in [HS81]. It relies on the hypothesis of brightness consistency, which states that the brightness of a given point in the scene will not change across time. It also assumes that the flow is spatially smooth. These two constraints are expressed into two energy terms of an optimisation problem, which is solved to supply the optical flow. A local optimisation approach still relying on the brightness consistency assumption was first designed in [LK+81]. The reader is referred to [FBK15] for a survey on optical flow estimation before the advent of deep learning methods.

Recently, the investigation of deep neural networks for optical flow estimation has allowed to outperform classical methods, while providing fast methods. In [Dos+15], the authors first showed the feasibility of applying deep learning methods to optical flow estimation. In [Ilg+17], the authors obtained state-of-the-art results comparable to classical methods, the latter being however much slower. The architecture used by the authors does not only involve stacked convolutional layers. It also include image warping with intermediate optical flow, and the use of a correlation layer between feature maps to help estimate the motion field. A sub-network specialised in small displacements is also designed, and the training schedule has been carefully selected.

A key element making the training of deep networks successful is the availability of large labelled datasets, and the task of optical flow estimation is no exception to this. Then, research efforts do not only focus on the development of new architectures, but also on the constitution of large training datasets. In [May+18], the authors discuss what makes good training data for optical flow estimation. For this problem, synthetic data is much easier to generate. The authors argue that realism is not as strong a prerequisite as it may seem, and that training with simple synthetic data with a data augmentation step is often sufficient. In fact, it has become a common practice to pre-train networks on Flying Chairs [Dos+15] or FlyingThings3D [May+16] (see Figure 2.7), which are both very unrealistic. The first one is generated by deforming chairs with affine transformations in 2D images, and the second by randomly moving 3D models of random objects in a 3D space. The authors of [May+18] emphasise that an important point is rather to carefully define the training schedule. In particular, simpler data should be presented first to the network to let it learn basic concepts. More difficult data should be presented only after this initial step. The authors even warn that training solely on challenging data yields poorer performance than training only on simpler data.

Many methods have since been developed to estimate optical flow with deep networks. In [XRK17], the authors estimate the flow with the help of a cost volume. To compute the cost volume, a deep network that embeds each pixel into a feature vector is trained. The idea of the cost volume is to represent likely displacements with low values and unlikely displacements with high values. For this, the cost volume is computed with dot products involving the features from the past and present images. The authors then apply a classical processing to estimate the flow.

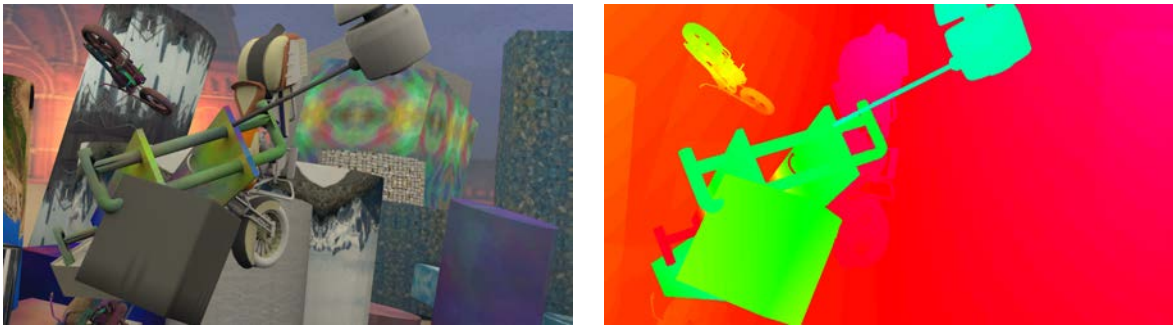


FIGURE 2.7 – Image from the FlyingThings3D dataset of [May+16] (left) and corresponding ground truth optical flow (right). This dataset consists of random objects in motion in a 3D space. As can be seen in the image, the objects are even allowed to pass through each other, which leads to a rather unrealistic scene. However, object motion is well defined in this dataset, which has proven to be relevant for the training of optical flow estimation methods.

Other methods chose to follow an end-to-end approach. For LiteFlowNet [HTL18] and PWC-Net [Sun+18], the authors notably rely on feature warping instead of image warping. In [HTL18], a flow regularisation layer is also included in the architecture to tackle the problem of outliers and vague flow boundaries. Both methods rely on a significantly more lightweight model than FlowNet 2.0, while achieving comparable or even better performance for some settings. Improving the training procedure for the same architecture is another way to improve results. This is what is done in [Sun+20] (already mentioned in Section 2.1.2) or in [BHW20]. In [BHW20], the authors modify the sampling procedure to ensure difficult fast motions are not under-represented in the training data. They also choose to decrease data augmentation and regularisation as training progresses.

Unsupervised learning is applicable to optical flow estimation [YHD16 ; ZN17], or even in the context of bio-imaging [LF18]. A common way is to use the brightness consistency assumption to define the training procedure of the network. The image is warped using the predicted flow, and the warped image is compared with the observed image. If the flow is predicted properly, the two images should be similar. Further refinements can be developed, such as the inclusion in the loss of a term ensuring the smoothness of the prediction. In [Liu+19], occlusion is also handled. They leverage non occluded areas, for which the flow can be more reliably obtained, by generating artificial occlusion in these areas.

The research field of optical flow estimation has seen many evolutions in recent years. The interested reader who wishes to get a deeper understanding of this domain is referred to the very recent survey given in [HR20]. Concerning our problem of motion saliency estimation, we will retain that we dispose of a way to estimate motion between frames almost in real time, and with a good accuracy.

2.5 Saliency and anomaly detection

Now, we will address saliency estimation in general, as well as anomaly detection. These domains are related to our objective of motion saliency estimation. We will review work concerning saliency based on eye-tracking in Section 2.5.1, static image saliency in Section 2.5.2, video saliency in Section 2.5.3 and anomaly detection in Section 2.5.4.

2.5.1 Eye-tracking saliency

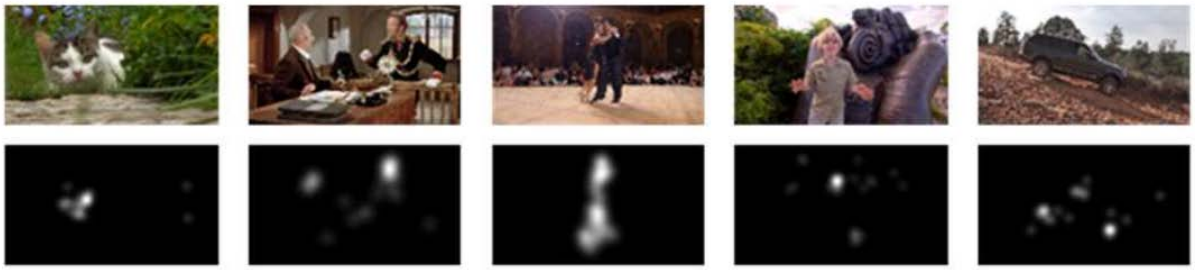


FIGURE 2.8 – Eye-tracking saliency maps (bottom) and associated frames (top) of the LEDOV dataset. The saliency maps are obtained from the recorded eye-fixation of 32 subjects, which are combined and smoothed. These samples illustrate that the gaze is first attracted by people, by faces and by moving objects. Reproduced from [JXW17].

One way to define saliency is to ask people to view images or videos, and to record the places they look at. This recording can be made with dedicated devices, as done for instance by [PLN02], [BT06] or [JXW17] (see Figure 2.8 for an illustration). People can be given a specific task when viewing to orient their attention. They can also be asked to watch with no particular instruction to observe where their attention will naturally focus. After the recording session, a succession of fixation points in the image is available for each viewer. It is then possible to define a saliency value for each pixel, by combining for instance the observations of all the viewers, and with a smoothing procedure to convert the fixation points to a two-dimensional valued saliency map [LMB13].

Many methods have been designed to solve this task. An example is given in [Liu+14]. In this article, the authors divide the image into superpixels. Then, they work at the superpixel-level to extract appearance and motion information, such as histograms of color and motion. Elements which attract the attention of the observer have a distinctive appearance and motion, which is used to build the saliency map. It is also common to include in the method a mechanism to reproduce the phenomenon of central bias, such as for instance in [PLN02]. Indeed, human observers tend to look more often at the central areas of images. Moreover, it is worth noting that the area people will look at will depend from factors such as their state of mind, their

interests of their age. These factors can consequently be exploited for a more precise saliency prediction. This is done for instance in [LM+17], in which the authors propose a saccadic model that is age-dependent. The goal of saccadic models is to estimate visual scanpaths, providing information about how people explore the scene.

A shift of paradigm has recently occurred, with more and more methods relying on deep neural networks to obtain state-of-the-art results. In [JXW17], the authors specifically built a large eye-tracking dataset for the training of such methods. They design a deep network with two branches to extract objectness and motion information, that are eventually concatenated. To predict the saliency map, the features are fed to a convolutional LSTM (long-short term memory), which is a form of recurrent neural network.

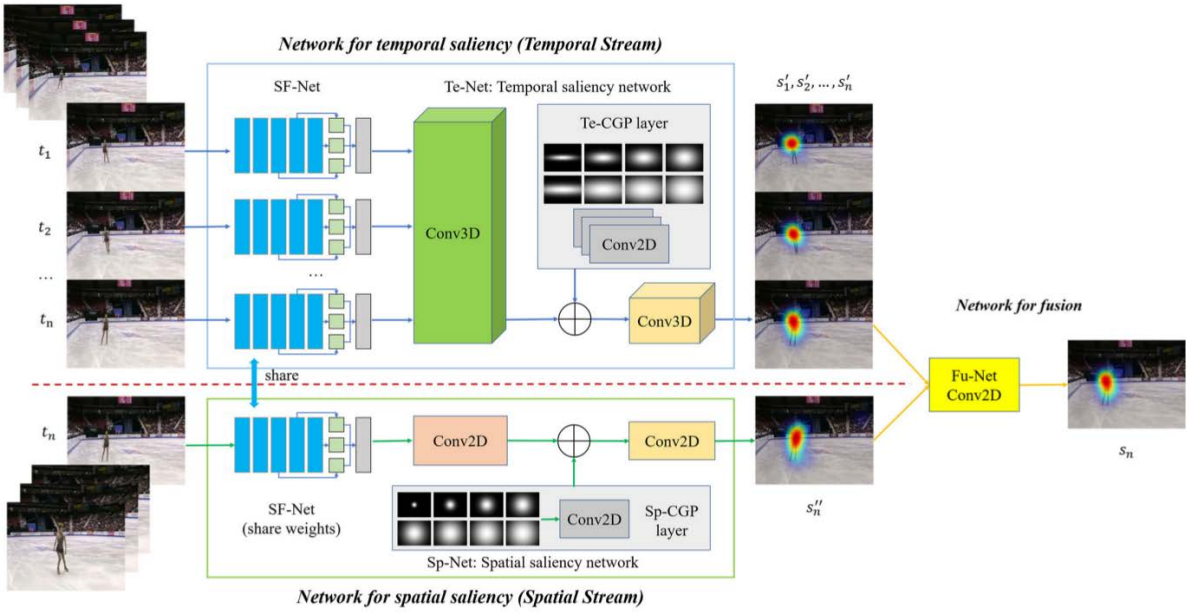


FIGURE 2.9 – Two-stream network for eye-tracking saliency proposed by [ZC18]. The temporal stream relies on 3D convolutions to process frames at different time instants. Reproduced from [ZC18].

In [ZC18], the authors similarly rely on a deep network with two streams to predict saliency (see the illustration in Figure 2.9). In their work, the temporal network relies on 3D convolution with 7 frames given as input simultaneously. For the training, they use the linear combination of three saliency metrics. By doing so, their objective is to obtain balanced performance. In [Lai+20], in addition to the two-stream network approach, the authors include a convGRU, which is a recurrent network to handle long-term dependencies. In [NSK20], the authors propose a network with a self-attention mechanism to handle images of human crowds.

Although deep networks can obtain good performance in general, exploiting the knowledge of the problem at hand is an effective way to improve results. In [LH18], the authors rely on a

spatial LSTM to build a deep spatial contextual long-term recurrent convolutional network. Their idea is that a spatial LSTM is a way to reproduce the behaviour of the human visual system, in particular the cortical lateral inhibition. The authors of [Wei+19] consider the task of predicting saliency maps for people with autism spectrum disorder. Knowing where autistic people look at can help design visual content such as teaching material that is more adapted to them. To solve this problem, the training of the network is adapted, with an initial pre-training stage on a dataset of standard observers, followed by a fine-tuning stage with data from autistic observers.

For a more detailed review of the literature of eye-saliency estimation, the interested reader is referred to [Bor18]. Finally, let us conclude this section by mentioning the work of [QGH18]. In this paper, the objective is to segment the foreground object in videos. To do this, the authors rely on eye-fixation information to provide the necessary clues to their framework. This is possible because objects in the foreground often attract the attention of observer. In fact, segmenting the objects in the foreground of images or videos is an alternative way to define saliency, that we will explore in the following section.

2.5.2 Static image saliency

FIGURE 2.10 – Static images and associated ground truth saliency from the MSRA-B dataset [Wan+17]. The ground truth is defined manually and is binary. The salient area corresponds to the object with the most distinctive appearance is the scene, usually an object in the foreground.

This section is concerned with static image saliency. When searching for saliency in an image alone, motion information will obviously not be taken into account. However, saliency clues present in static images still remain in videos. Then, they may be relevant for video saliency as well. Therefore, we will distinguish methods for video saliency in general and methods for video *motion* saliency in particular.

Ground truth saliency in images can be defined by the mask of the foreground object with the most distinctive appearance. Examples are given in Figure 2.10. With this definition, saliency has a binary value for each pixel : either it is salient or it is not. In fact, it can be assimilated to object segmentation, the object to be segmented being the salient object. This is in

contrast to eye-tracking based saliency, for which there is no explicit notion of object and for which even the ground truth is valued. Let us note that this distinction affects mainly the ground truth. In practice, many methods will predict valued saliency maps even with a binary ground truth.

Although defining saliency that way is not equivalent to resorting to eye-tracking data, there are many common points. In both cases, an object with a very distinctive appearance compared to the background is likely to be salient. In [Jia+13], the authors begin by grouping pixels into superpixels to get a simplified representation of the image. Then, they consider the graph formed by the superpixels, which is processed with the aid of Markov chains to extract saliency.

Another typical approach is to design features, which are supposed to characterise saliency. Examples of features include color, texture, shape or brightness. In [LLU20], these features are leveraged to find saliency in microscopy imaging. In [EE13], the authors also consider such features but go one step further, by combining them into covariance matrices used in turn as descriptors for saliency estimation.

In [JH13], the authors consider that low-level saliency features are not sufficient, and they design a method that relies on objectness estimation. An object is represented by a bounding box, which is a rectangle designating the object location. Candidate bounding boxes are then evaluated with dedicated criteria. For instance, the borders of the object are supposed to be associated to strong edges, and the colors of the object are supposed to depart from the colors of the surrounding. The object locations are refined to obtain the final saliency map with a Markov random field.

Again, the recent years showed a shift of paradigm, with more and more works leveraging deep networks.

In an early attempt [LY16], the authors used a deep network to extract deep features at several scales for saliency estimation. The network is trained on the large ImageNet dataset [Rus+15] initially defined for object classification. A very large amount of data is required to train a deep network, and many more training images are available for object classification than for saliency estimation.

In [Zen+18], the authors also rely on a network pre-trained for image classification. They argue that saliency is image-specific, as the same object may be salient or not, depending on the image. Then, they do not predict directly saliency. Instead, the network is fine-tuned, not to produce saliency labels directly, but to estimate a similar embedding for all pixels belonging to the salient region, and another embedding for pixels belonging to the non-salient regions. At test time, an existing saliency detector provides an initialisation for the salient regions. This detector is then “promoted”, that is, the embeddings estimated by the network are used to iteratively refine the original saliency map predicted by the detector.

In contrast to works cited above, the authors of [LH16] go one step further. They propose

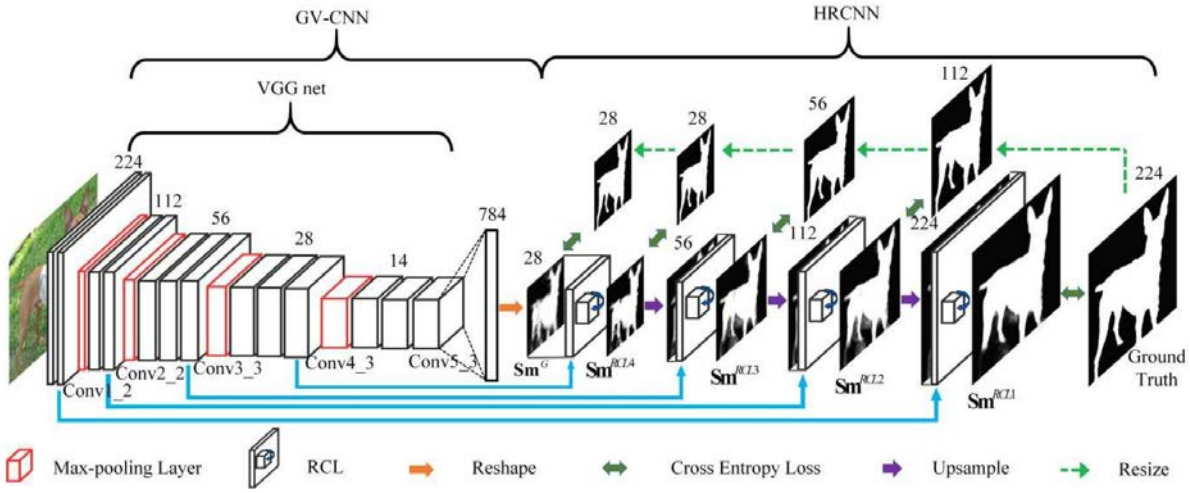


FIGURE 2.11 – Convolutional neural network for static image saliency from [LH16]. The network is composed of a contracting and an expanding part, which is typical of a U-net architecture [RFB15]. This structure allows one to integrate information from distant areas in the image, while limiting the computational burden. For the method of [LH16], the loss is applied not only to the final saliency map, but also to lower resolution intermediate saliency maps. Reproduced from [LH16].

a network called DHSNet, which is an end-to-end architecture : the network predicts the saliency map from an input image with no additional processing (see Figure 2.11). Designing their method this way allowed the authors to reach a speed of 23 fps on GPU. To predict saliency, the authors estimate a coarse saliency map in the first part of their network. Then, this map is refined with a hierarchical processing including recurrent convolutions.

Due to the promising results and the high computation speed of end-to-end approaches, many other works have explored this approach for saliency computation. With end-to-end networks, the focus shift from the design of the features to the design of the network. In [Hou+19], short connections are introduced, which create shortcuts between different layers to provide better multi-scale features. In [Zha+18], recurrence and attention are combined to estimate saliency. Attention consists in introducing a mechanism in the network to associate a multiplicative weight to features, for instance between 0 and 1. The higher the weight associated to a given feature, the more the attention of the network will be focused on this feature. The attention module can act channel-wise to select the most relevant feature, or spatially to highlight the most relevant regions. The authors argue that attention mechanism is beneficial for saliency, to avoid distraction from irrelevant details.

2.5.3 Video saliency

This section about video saliency is organised in two sub-parts. First, methods whose objective is to estimate generic valued saliency maps will be presented. Then, we will consider the video object segmentation task, whose objective is to identify the most distinctive objects in videos, and which is somewhat related to video saliency.

2.5.3.1 Valued video saliency maps

Video saliency can be defined by extending image saliency to videos. For video saliency, motion will play a role to determine which elements should be salient in the ground truth. The salient elements may still have a distinctive appearance compared to the background, but they are also supposed to undergo distinctive motion. Consistently, video saliency methods are generally designed to leverage both appearance and motion.

In videos, apparent motion can be caused by two factors. First, objects in the scene may be moving in the 3D scene. Second, the camera itself may be moving. Depending on the position of the objects in the 3D scene, they will have a specific apparent motion. Methods such as [Fan+14 ; KK14 ; MGS14] directly estimate saliency from appearance and motion cues. Other authors, for video saliency map estimation [Hua+14], but also for eye fixation saliency estimation in videos [LLB07], first compensate for the camera motion. In both cases, the scene may comprise dynamic textures [Cri+13] such as flags in the wind or ripples on the water, which are generally not considered as salient and that will act as distractors.

The problem of video saliency map estimation is emblematic of the paradigm shift that followed the generalisation of deep learning. Earlier methods are classical methods [Kim+15 ; WSS15 ; WSP15 ; Kar+16 ; LS16 ; Guo+18 ; Ayt+18 ; Che+18a], which rely on expert knowledge to solve the task. Appearance is leveraged for instance through the use of intra-frame boundary information and contrast [WSS15], object proposals [Guo+18], or color contrast and background cues [Ayt+18]. Motion information is exploited by relying on motion boundaries [WSP15], by enforcing temporal consistency [Guo+18], or by relying on motion contrast and temporal superpixels [Ayt+18]. The final saliency map can be obtained through the use of graphs [Kar+16], by relying on a hierarchical structure of the elements in the frame [LS16], which is a common processing step to better handle scale variations, or with statistical tools [Che+18a]. The method of [LS16] is further illustrated in Figure 2.12.

Many recent methods rely instead on deep networks to predict saliency. In [LS18], the authors use spatio-temporal deep features extracted with a deep network to estimate video saliency. They extend conditional random fields (CRF) to the temporal domain to refine the saliency map. Other works propose more straightforward architectures. In [WSS18], the authors use a network that extracts first static saliency from single images. Then, dynamic saliency is

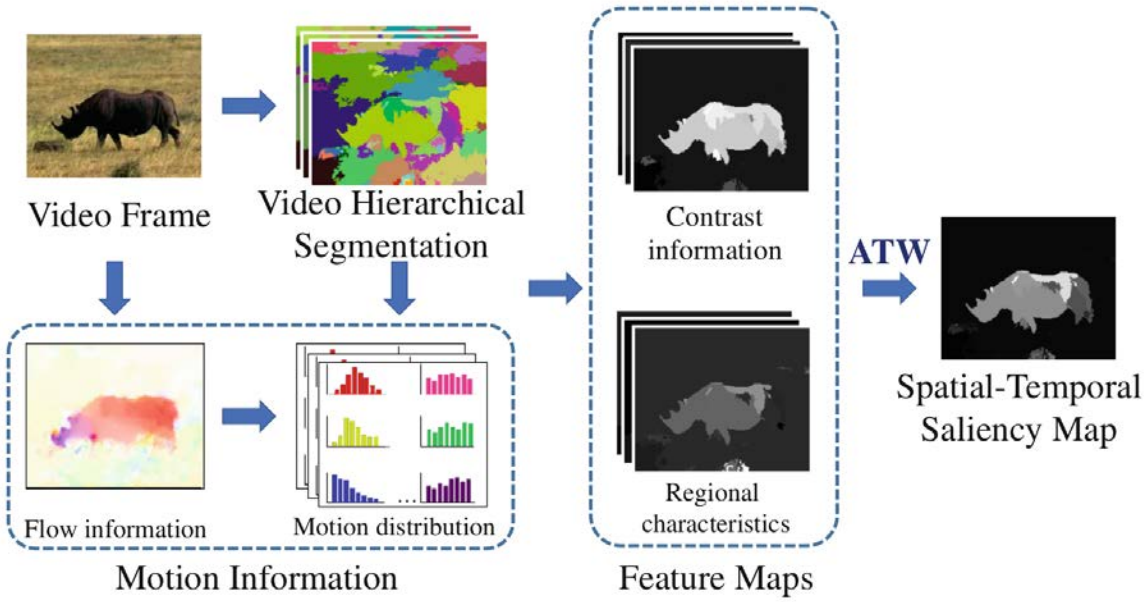


FIGURE 2.12 – Method of [LS16] for video saliency. The authors rely on appearance to realise a segmentation of the video, and on the optical flow to extract motion information. The saliency map is obtained by combining the spatial and temporal information, with the help of Adaptive Temporal Window (ATW). Reproduced from [LS16].

estimated with image pairs concatenated to the static saliency map. In [Li+18a], the authors introduce a recurrent LSTM-based network that also uses feature warping with the optical flow to compensate for object motion. Finally, the authors of [Che+20] identify long-term saliency clues. The temporally distant regions are spatially aligned with the current region of interest, and a deep network uses the aligned regions to predict saliency.

2.5.3.2 Video object segmentation

FIGURE 2.13 – Samples and ground truth from the DAVIS 2016 dataset [Per+16] for Video Object Segmentation. The salient element is a moving object in the foreground of the video.

Finally, video saliency has several connections with the Video Object Segmentation (VOS)

task. The latter consists in segmenting through the video the foreground moving object of interest (see Figure 2.13 for an illustration).

The first variant of VOS is unsupervised VOS. In this particular context, the term unsupervised refers to the fact that no specific indications are given regarding the location of the object to segment. Methods can again be divided between classical ones [PF13 ; FI14 ; JKK17 ; HHS18 ; Zhu+19] and deep learning-based ones. In [PF13] and [FI14], the authors rely on appearance and motion cues. In [JKK17], the authors use superpixels to generate candidate regions and to merge them. They also explicitly rely on the prior that there is one primary object, present in the whole sequence. Edge cues are exploited in both [JKK17 ; HHS18], and they are used by [HHS18] as part of a neighbourhood graph. In [Zhu+19], the authors combine motion saliency and object detection. They include several refinements, based on CRF, region fusion and long-term combination.

A first example of deep learning-based method is [DJXG17]. It consists in a two-stream CNN, with a video frame and optical flow as input. The authors use weakly annotated videos and image recognition datasets to augment the amount of training data. In [TAS17], a deep network leverages optical flow to find objects in motion. In [Son+18], the authors propose a deep architecture with a bidirectional convLSTM and a pyramid structure to handle multiple scales. They include a training stage on static images for data augmentation too (see Figure 2.14 for the workflow of this method). In [SHT20], the authors propose a non-local deep network, designed to capture global dependencies. In [Yan+19], the authors detect moving objects in videos with an adversarial setup. An inpainting network attempts to reconstruct the optical flow inside a mask from the flow outside of it. A generator network attempts to estimate the mask that would make this reconstruction difficult, which can be achieved by predicting masks corresponding to moving objects. This work was developed in parallel to our work that will be presented in Chapter 3, and that will rely similarly on optical flow inpainting.

The task of VOS is also declined in a semi-supervised setting. In this case, the location of the object to track in the first frame is known. The authors of [Wan+19b] adopt a super-trajectory representation. A super-trajectories corresponds to a set of point trajectories, which are grouped because these points share a consistent motion pattern, a similar appearance and they are close spatially. In [Xia+18], a deep network is trained in two parts : a first offline pre-training, and an online fine-tuning on the first frame of each video to process. This is typically done to leverage the location information provided in the very first frame of the video, at the expense of an increased computational load at test time. In addition, the optical flow is used to warp and align the representation of successive frames. In [Yan+19], the authors generate pseudo-labels from sparse annotation as a core component of their method. This is a way to reduce the dependency on dense labelling and to get a high amount of training data.

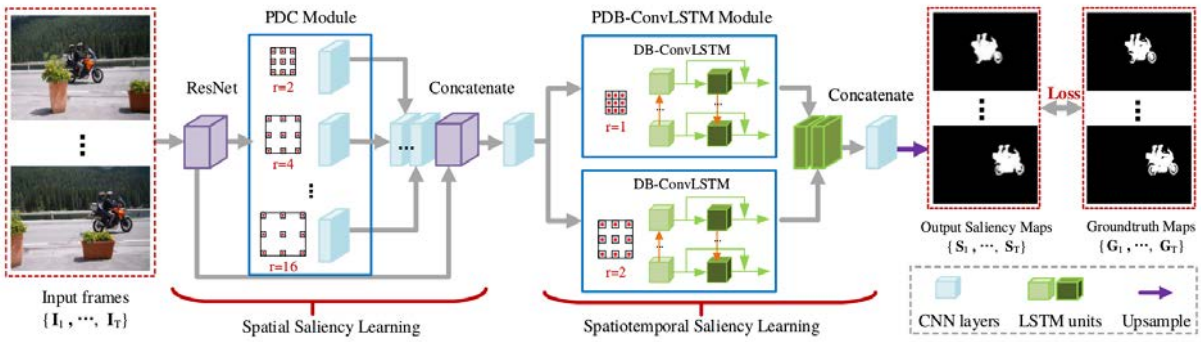


FIGURE 2.14 – Deep network designed in [Son+18] for Video Object Segmentation. A notable characteristic of this network is the pyramidal processing in the spatial part to better handle multiple object sizes. The authors also rely on recurrent networks to exploit the temporal dimension. Reproduced from [Son+18].

2.5.4 Anomaly detection

Anomaly detection consists in identifying elements that differ from the norm, the norm being defined by some models or simply corresponding to what is observed most of the time. Examples of anomalies include people running in an usually quiet place, manufactured objects with unexpected shapes, or suspect log data obtained when monitoring an ICT (information and communication technology) system.

Anomalies can be linked to motion, as for the detection of anomalies in videos of crowded scenes. To solve this problem, the authors of [DC10] rely on a representation of motion based on histograms of directions combined with a separate indication of the speed. In [PRBB17], the authors estimate local motion models that are combined into local histogram descriptors. They introduce a dedicated distance to assess how much these histograms differ to a representation of normality, which allows them to identify anomalies. Abnormal motions are searched for in [Man+11] following a multi-scale approach that leverages optical flow features. In [DDM19], the authors are concerned with the detection of small anomalies in moving background. To detect them, they exploit optical flow for background subtraction. Then, only noise and potential anomalies are left, and the distinction between the two is achieved with a statistical framework. In [Bar+19], the authors take a broader view, by considering anomalies in time series data in general. They define a framework to compare efficiently segments of different length. For estimating the similarity between samples, they define a variant of the Kullback-Leibler divergence.

The methods quoted above are all based on classical processing. Recently, deep networks have been investigated for anomaly estimation. First, auto-encoder networks, as the one represented in Figure 2.16, have been adopted. The training objective of an auto-encoder is to reconstruct its input as precisely as possible. To avoid the trivial solution which consists in

learning the identity function, the input is generally compressed to an intermediate compact representation, also called the latent representation or latent code. In [RLL18] and in [MYY18], the authors decide that, during the training stage, the auto-encoder only sees normal data. Then, they assume that once trained, the network will be able to reconstruct properly only normal data and not anomalous data. The reconstruction error of the auto-encoder serves to detect anomalies. This approach was applied to video saliency and defect inspection on textured surfaces respectively. In [Rav+17], the authors leverage the GAN framework (Generative Adversarial Network, [Goo+14]) to reconstruct the image from the optical flow and vice versa. They still expect that only normal events are seen among the training samples and will be properly reconstructed.

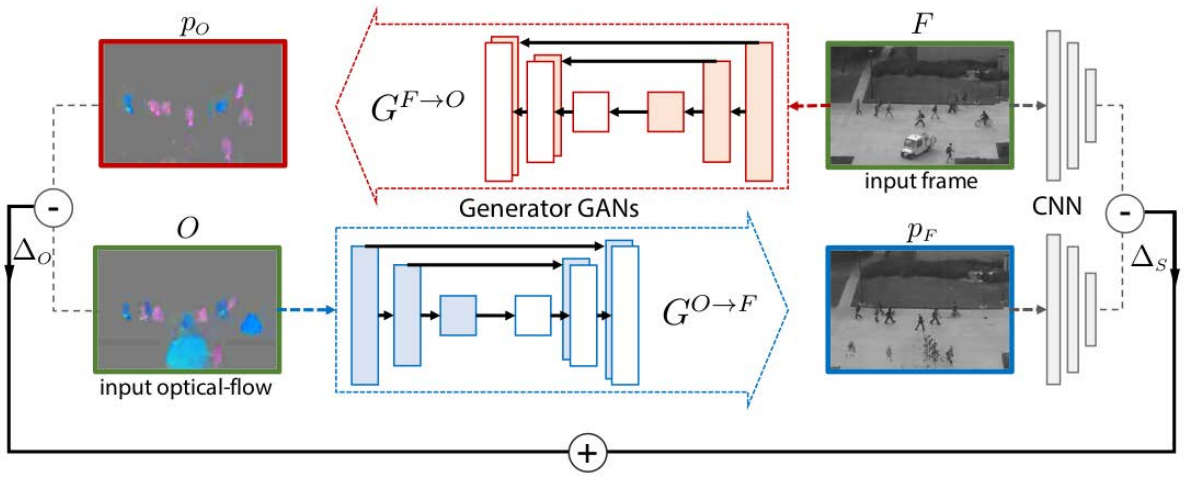


FIGURE 2.15 – Method of [Rav+17] for abnormal event detection. A GAN (Generative Adversarial Network) is trained to predict normal image and flow from normal flow and image respectively. Reproduced from [Rav+17].

Another family of methods uses auto-encoder and GAN frameworks, but focuses on the latent representation. In [Xu+17], the authors choose denoising auto-encoders to learn features from images, optical flow, and a combination of both with early fusion. The final anomaly decision relies on classical one-class SVM classifiers [Sch+01]. Other methods enforce specific properties for the representation thanks to an adequate training stage. In [Ghe+19], the authors train a GAN in order to represent normal data with codes following a normal distribution, and anomalous data with codes outside of this distribution. In [Wu+20], an auto-encoder is trained that ensures the following property of the latent representation : a point between two latent points, representing both normality, should be also a plausible representation of normality. Anomalies codes should then lie in a different manifold than normal samples.

In more specific applications, anomalies can be characterised with a precise model. This is the case in biology and physics for the study of the diffusion (the motion) of particles. Anoma-

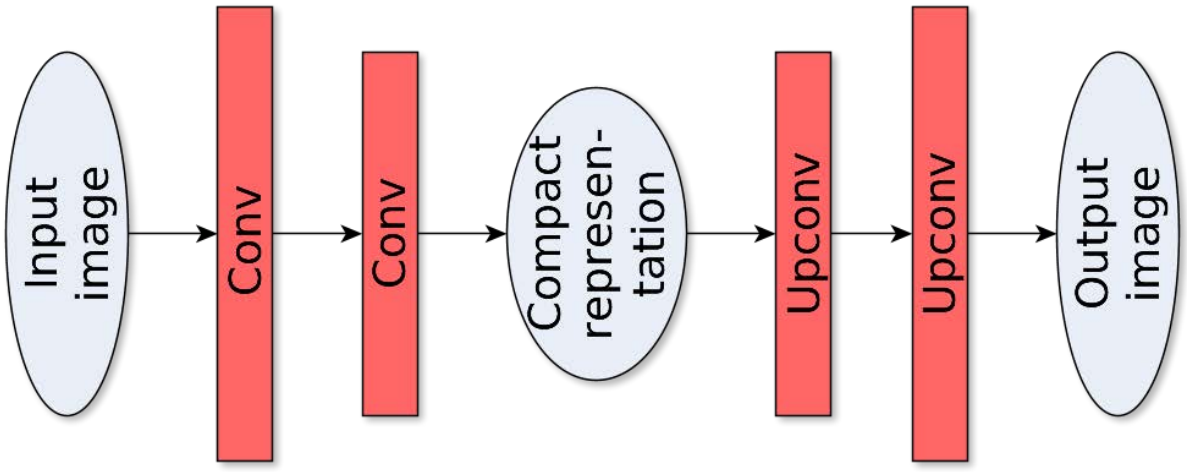


FIGURE 2.16 – A basic image auto-encoder network. The input is processed by the encoding part of the network into a compact representation. This representation is then decoded by the decoding part of the network, whose role is to reconstruct the input. The small dimension of the compact representation acts as a bottleneck that prevents the network to trivially learn the identity function.

lous diffusions correspond to motions that are either faster or slower than expected. Identifying anomalous diffusion is interesting to identify physical and biological phenomena. In [Bo+19], the authors estimate the parameters of the diffusion with a recurrent neural network. From these parameters, it is then straightforward to detect anomalies.

Let us conclude this section by mentioning the survey on anomaly detection with deep learning [CC19]. One interest of this survey is the opposition made by the authors between deep hybrid models and one-class neural networks. Deep hybrid models [AMG16 ; EMK17] use a network to learn a representation, and classify it with a classical algorithm. On the other hand, one-class neural networks [Ruf+18 ; CMC18] involve an end-to-end training to find anomalies. The network is expected to form an hyperplane or hypersphere to separate normal data from anomalies. The authors argue that an end-to-end training gives one more control over the training process, which is expected to improve performance. Let us emphasise that this distinction is made possible because, for the anomaly detection task, there is often an absolute definition of normality. Normality corresponds to the behaviour observed most of the time, for instance pedestrians walking quietly, and not doing unexpected actions such as randomly throwing objects. Anything that deviates from this norm is anomalous.

If we come back to our objective of motion saliency estimation, one task of interest is to estimate relative motion saliency. It consists in finding salient motion among non-salient motions (for instance a crowd going in one direction, and a salient individual moving in the opposite direction). This will be addressed in Chapter 5. We can already stress that in this case, it does

not make sense to define saliency in an absolute way : a pedestrian moving to the left is salient in a group of pedestrians moving to the right, but not in one moving to the left.

2.6 Trajectories

Optical flow provides information about instantaneous motion. However, motion saliency can appear progressively over time. In addition, the optical flow is more difficult to exploit when the objects are denser in the scene, and it becomes harder to distinguish motion patterns. If the area of interest is very large, it may simply not be easy to cover it with a single camera. Trajectory representation is then an attractive alternative in these situations.

2.6.1 Trajectory extraction

The simplest way to get trajectories is to measure them with relevant sensors such as GPS, as in [End+16]. With the growing number of devices equipped with this technology (cars, but also mobile phones), it provides an opportunity to analyse and better understand motion patterns either in roads or crowds. Autonomous driving vehicles need to analyse their surroundings in order to navigate successfully. In this case, more complex sensors such as laser scanners [Gid+10] followed by an adequate processing step can be used to detect road users such as pedestrians and to extract their trajectories.

When a direct measurement is not possible, it is common to estimate trajectories from videos with a tracking algorithm. Methods such as the KLT tracker [LK+81 ; TK91] use optical flow to build trajectories. The algorithm includes a step to search for points with distinctive enough features, for which the optical flow should provide the motion with no ambiguity. Specific trackers can also be developed depending on the domain, and in particular depending on the imaging technique used. In [CBO13], a tracker is specifically designed for biological imaging. The goal of this tracker is to handle thousands of cells in a cluttered environment, and the authors adopt a probabilistic framework to achieve it.

To collect trajectories over a large scene, one can design a large-scale tracking system such as the one used in [ARF14]. To analyse displacements of people in a train station, a network of cameras captures the scene with an image processing pipeline involving re-identification.

The family of Multiple Object Tracking (MOT) methods, such as [Chu+17 ; Zhu+18 ; MCA19], tracks multiple objects in videos. In [Chu+17], the authors combine a single object tracker and an attention mechanism in a deep network to extract the trajectories. In [Zhu+18] attention is also exploited and a tracking loss is defined to focus on hard negative samples. Hard negatives typically correspond to people with a similar appearance compared to the target, that should not be confused with it. A data association step enables to recover from temporary failures

of the tracking, for instance due to occlusion. In [MCA19], the authors propose a tracking by detection framework that relies on the processing of an interlaced representation of the video by a dedicated deep network.

To conclude this section, we refer the reader to [DT14] for a survey of classical tracking methods, and to [Cia+20] for a survey of Multiple Object Tracking with deep learning approaches.

2.6.2 Trajectory-based video analysis

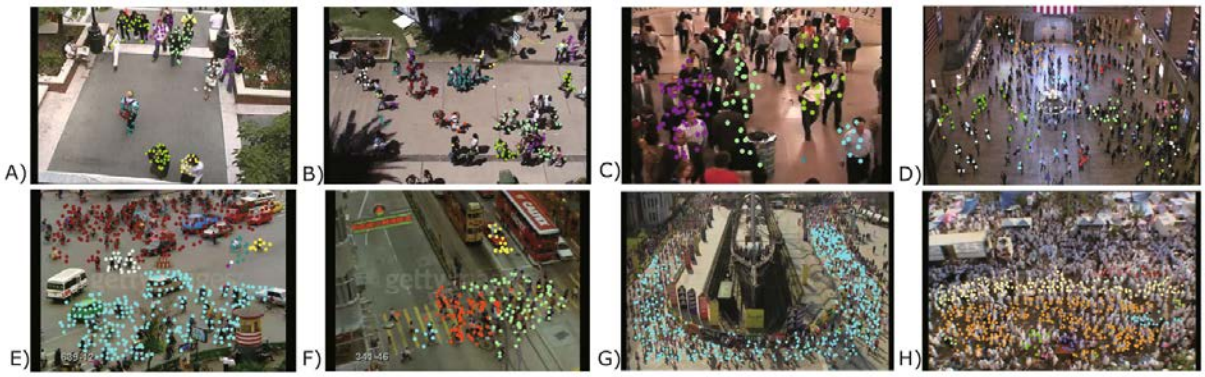


FIGURE 2.17 – Motion analysis from trajectories with the method of [ZTW12]. Coherent motions are clustered, which provides information regarding the scene. In the images, the dots with the same color represent trajectories belonging to the same cluster. Reproduced from [ZTW12].

Trajectory data can be used for a variety of tasks. An obvious use case is for motion analysis in videos. In [ZTW12], the authors exploit trajectories to analyse dense crowds of people. They propose a trajectory clustering algorithm to detect areas with a coherent motion. Similarly, the authors of [SLW14] detect groups in crowd scenes by clustering tracklets extracted with the KLT tracker. In [PR+16], the authors rely on trajectories to construct a hierarchical representation of trajectories in a scene. This representation is obtained by a sparse-coding scheme and by applying tree-clustering to the codes. The different levels of the resulting hierarchy go from the most general (the camera motion) to the more specific (motion of the object, then motion of the object parts). The authors of [BFM19] are also investigating motion characterisation, and they propose an unsupervised hierarchical multiple motion model. It allows them to represent and classify different types of agents such as pedestrians and bikes. Trajectories can be used as well to characterise a unique element, as done in [GGCR17]. The authors consider trajectories of facial landmarks of people while driving. The goal is to automatically detect and raise a warning when the attention of the driver is oriented toward a smartphone rather than toward the road.

Regarding the biology imaging field, a fine analysis of trajectories in tissues can lead to an improved understanding of the underlying mechanisms. Motion of particles can be characte-

rised for instance by a classification into Brownian, sub-diffusive and super-diffusive classes. In [Bri+20], the authors achieve such a classification with a statistical framework. They identify confinement domains in cells. In [Gra+19], the authors make use of a deep network to classify the diffusion of particles into Brownian, fractional Brownian and continuous time random walk. They also estimate parameters such as the Hurst exponent and the diffusion coefficient, which characterise these kinds of motion.

Going back to natural images, long-term motion to segment objects has been explored by several works. In [OMB14], the authors note that short-term motion may exhibit inconsistencies. For instance, an animal may be static for a given period of time, and the legs of a walking person display a different motion than the body. They argue that trajectories represent long-term information that can address these concerns. In [KAB15], the authors also rely on trajectories for object segmentation in videos. They build a long-term point trajectory graph from the video sequence. From this graph, they formulate an optimisation problem, which allows them to cluster the trajectories into objects.

Trajectories have also been used to find parts of images that attract the visual attention due to their motion. In [Cam+17], the authors make use of the Hough transform applied to motion, to group motions based on the salient attractive zones where they converge. Besides, in [WQT15] the authors address the action recognition task with deep features. They extract trajectories that are then used to pool the deep features, in order to get relevant descriptors.

Computing similarity between two trajectories is useful in various contexts and several classical metrics have been proposed for this, such as Dynamic Time Warping [YJF98] or the Hausdorff distance [AMP10]. In [Yao+19], the authors speed up the computation of known metrics by approximating them with a deep neural network. Still, recent methods use deep networks to extract a compact representation, in particular with auto-encoders [Yao+17 ; Su+16 ; CR+18 ; Cho+18]. In [Yao+17], this representation is used for trajectory clustering, in [Su+16] for crowd scene understanding. It is included in a reinforcement learning framework in [CR+18].

Finally, let us consider trajectory anomaly detection, which is closer to our goal of motion saliency estimation. In [JH96], a statistical model of trajectories is learnt to recognise events and to detect anomalies. In [HBL08], for the same objective, local differential features, invariant to transforms such as rotation, translation and scaling, are used. These features are embedded with a hidden Markov model. In [LF14], the authors develop an algorithm for trajectory anomaly detection involving online learning, allowing to incrementally augmented the training set.

For trajectory anomaly detection, recent methods again use deep networks. In [RB18], the authors deal with road user trajectories. They reconstruct them with auto-encoders. They assume that trajectories poorly reconstructed are salient, as salient trajectories are rare in the training data. In an extended work [RB19], they use a Generative Adversarial Network (GAN) to estimate saliency. They train the discriminator to distinguish normal and abnormal recons-

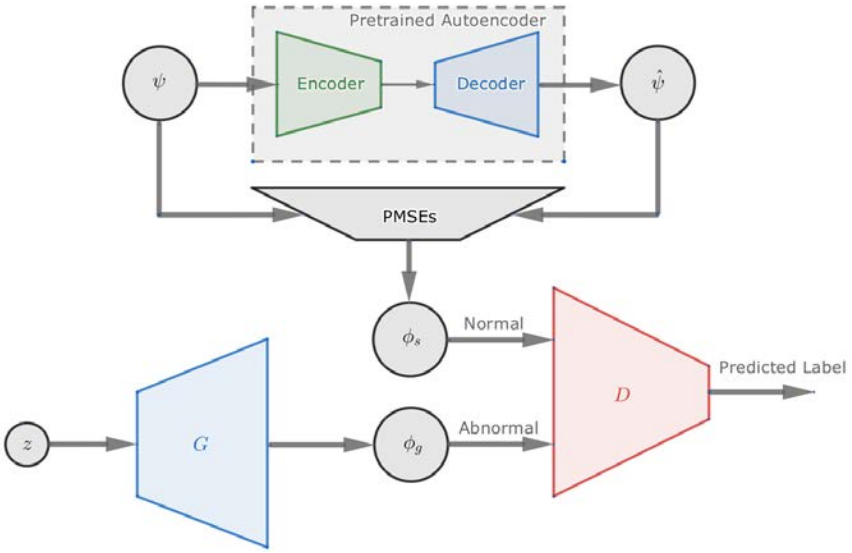


FIGURE 2.18 – Method of [RB19] for abnormal trajectory detection. The core of the method is an auto-encoder trained with normal trajectories that is expected to reconstruct correctly only normal trajectories. A Generative Adversarial Network (GAN), composed of a generator G and a discriminator D, is used to distinguish reconstruction errors from normal and abnormal trajectories. Reproduced from [RB19].

truction errors (see Figure 2.18 for the illustration of this method). The role of the GAN is to avoid the need to set manually a decision threshold, but the foundation of the method remains the reconstruction error.

The trajectory representation is not limited to moving objects, as it can be used for time series data in general. In [Fan+18], the authors consider building energy data over time. They base their anomaly detection method on the reconstruction error of an autoencoder. In [Wan+19], the authors propose a method for generic time series classification. The combination of a convolutional neural network and an adversarial regularisation scheme allows for a better interpretability of the decision.

2.7 Conclusion

In this chapter, we have seen how saliency methods of the literature leverage both appearance and motion cues to highlight regions in images or videos. This can be done either with classical frameworks, or more recently with deep learning methods. In both cases, most methods mix appearance and motion cues to produce a unique saliency map. However, we cannot decipher if emphasis is put on an object because no other objects are similar in the scene, or because this object is moving in a singular way. In contrast to these approaches, we will focus

exclusively on motion saliency.

First, our work devoted to temporal motion saliency detection will be presented in Chapter 3. It will explicitly address the issue of whether motion saliency is present or not in each frame of the video sequence. This information is needed before attempting to locate saliency in images. However, existing methods usually ignore this aspect by assuming that saliency is present when computing any saliency maps. To develop our classification methods, we will leverage deep networks due to their successful application in many fields, and optical flow as an adequate representation of motion. The second method we propose for motion saliency map estimation will similarly rely on optical flow to extract regions that are salient due to their motion. We will take into account that saliency is context-dependent. This method will be detailed in Chapter 4.

To handle long-term and progressive motion saliency, we will use trajectories computed in videos. We will search for an adequate compact representation of trajectories, and the auto-encoder framework will be a proper basis for that. Methods for anomaly detection are more concerned with an absolute saliency value, for instance by relying on the reconstruction error of auto-encoders. As a consequence, they are not adapted to our problem. Instead, we will take inspiration from deep metric learning to propose a weakly supervised method for this task. We will develop further this subject in Chapter 5.

CHAPTER 3

FRAME-BASED MOTION SALIENCY DETECTION

The problem addressed in this chapter appertains to the well known domain of dynamic motion saliency in videos. However this is a new problem since we aim to first detect the temporal segments of the video where motion saliency is present. It turns out to be a frame-based classification problem. A frame will be classified as dynamically salient if it contains local motion departing from its motion context. Temporal motion saliency detection is relevant for applications where one needs to trigger alerts or to monitor dynamic behaviours from videos. It can also be viewed as a prerequisite before computing motion saliency maps. The proposed approach handles situations with a mobile camera. It involves two main stages consisting first in cancelling the global motion due to the camera movement, then in applying a deep learning classification framework. We have investigated two ways of implementing the first stage, based on image warping, and on residual flow respectively.

3.1 Introduction

Dynamic saliency comes to highlight objects undergoing separate, singular or unexpected motion in the video sequence. In the literature, dynamic (or video) saliency is mainly attached to a foreground object moving in front of a (mostly) static scene, as described in Chapter 2. Then, appearance may play a key role and is consequently exploited. In contrast, we take a slightly different definition of dynamic saliency. It involves all situations where local motion departs from its context with differences or not in appearance. Then, we will prefer to talk about motion saliency.

The specific problem addressed in this chapter is the temporal detection of motion saliency in videos, i.e., determining throughout the video which frames contain motion saliency. This enables to recover time intervals for which motion saliency is present in the video.

To our knowledge, this problem has not been investigated so far, but it is crucial for numerous applications. It can help detecting obstacle irruption for mobile robotics or autonomous vehicles, raising alert for video-surveillance, triggering attention for video analysis, or highlighting

relevant information for video summary. This frame-based classification acts as a pre-attention mechanism. More specifically, our method can be viewed as a prerequisite to the computation of dynamic saliency maps. The existing methods extract salient moving objects, while implicitly assuming that the frame is dynamically salient. More precisely, these methods supply valued motion saliency maps. One could consider that they could recognize a dynamically non-salient frame, by providing a "practically" empty output. Yet, it would require designing an additional stage to reliably decide it. Anyway, we experienced that motion saliency maps are hallucinated by existing methods on non-salient frames. This shows that existing methods are not able to address the frame-based motion saliency detection problem. Our method precisely addresses this issue.

As aforementioned, temporal detection of motion saliency in videos is a classification problem. It consists in deciding for every frame of a video sequence whether it should be labelled as dynamically salient or not, that is, whether it contains elements whose local motion deviates from its surroundings. In practice, the latter will be the global (or dominant) motion in the image. We will adopt the convolutional neural network (CNN) framework to solve this classification problem. Before classifying each frame of the video, we cancel the global motion usually due to the camera movement. We have investigated two ways of implementing it, based on image warping and on residual flow respectively. This leads to two main variants of our temporal motion saliency detection approach.

The chapter is organised as follows. In Section 3.2, we present the synthetic and real video datasets built to evaluate our classification scheme. In Section 3.3, we describe our approach and the two variants based on image warping and on residual flow respectively. Experimental results are reported in Section 3.4. Section 3.5 includes concluding comments.

3.2 Datasets

We start by describing the synthetic and real datasets used in our experiments. The dataset is indeed a key point for learning-based approaches, especially in the case of deep learning methods such as the ones we propose in this chapter.

3.2.1 Synthetic dataset

Machine learning methods, and especially deep learning methods, require a training set large enough for successful learning and generalisation [Sun+17]. Recently, it was demonstrated for computer vision applications [Dos+15 ; May+18], that learning can be efficiently achieved on synthetic datasets. We have then built a synthetic dataset for a first training stage of the networks involved in our methods.

Each element of the synthetic dataset consists of a frame pair. The second frame of the pair is obtained by applying to the first frame a parametric motion model. We take an affine motion model. Motion saliency is attached to added patches taken from other images and undergoing a different affine motion. Images of PascalVOC 2012 [Eve+12] were used to generate this dataset. Samples are provided in Figure 3.1. Frames are generated with a probability of 0.5 for absence of motion saliency, 0.25 for presence of one salient moving patch, and 0.25 for presence of two salient moving patches. Added patches have a limited size, of approximately 0.5% to 1.5% of the frame. We deliberately chose to include small-size patches to produce motion saliency examples hard to detect.

By generating salient examples that way, we could be faced with the risk that the network could detect salient frames thanks to the appearance of the added patches which are generally different from the content of the reference frame. To avoid this, non salient frames can also contain up to two added patches coming from unrelated images, but still undergoing the global motion. The shape of the patches extracted from other images was generated randomly, as illustrated in Figure 3.1.

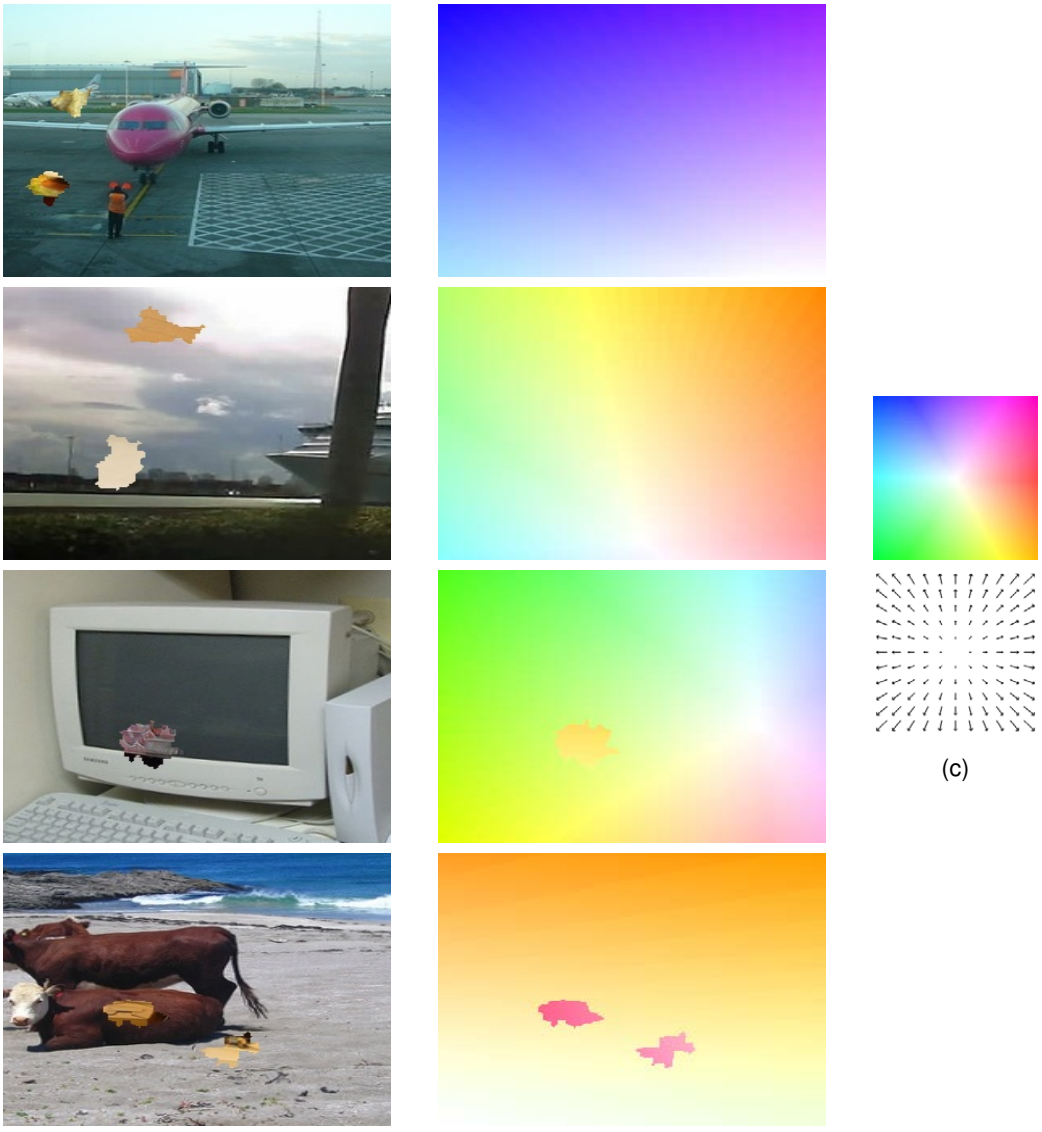
In order to maximise the variability of the training samples, frame pairs are generated on the fly during training. Approximately 4 million training samples are generated this way. The validation and test sets contain 2000 frame pairs each.

3.2.2 Dataset of real videos

A dataset with real videos is necessary for fine-tuning the networks and to assess the motion saliency classification methods. Since the proposed methods will have to handle videos with a mobile camera, the camera should be moving for a significant part of the dataset. Moreover, the ground truth must express the presence or absence of motion saliency at the frame level. The dataset constructed by [BL16] meets these requirements. It gathers FBMS-59 [OMB14], Complex Background [NHLM13] and Camouflaged Animals [BLM16], all of them being specifically re-annotated for the problem of moving object segmentation. This dataset contains difficult examples, in particular the videos of Camouflaged Animals for which motion information strongly prevails in the perception of the salient moving animals. We label a frame as dynamically salient in the ground truth, if it contains at least one independently moving object.

Since our objective is to detect temporal intervals of motion saliency in videos, non salient frames are required to serve as negative examples. Non salient frames are rare in the dataset of [BL16]. Consequently, we acquired 71 additional videos with no salient frames, i.e., depicting static scenes, but acquired with a mobile camera. These videos depict indoor, urban and natural scenes. The ground-truth is by construction immediately available for all the frames of the 71 additional non-salient videos.

The final dataset includes 144 videos, and is split in a training set of 94 videos, a validation



(a) (b) (c)

FIGURE 3.1 – Samples of the synthetic video dataset. (a) The top two frames are not salient while the bottom frames are salient. All frames include added patches cropped from another image, but only the patches of the bottom frames undergo independent motions. (b) The corresponding motion fields are represented with the colour code depicted in c). The optical flow is globally smooth in the top two rows, while it includes outlier patches in the bottom two rows. (c) Colour code for the flow field represented below.

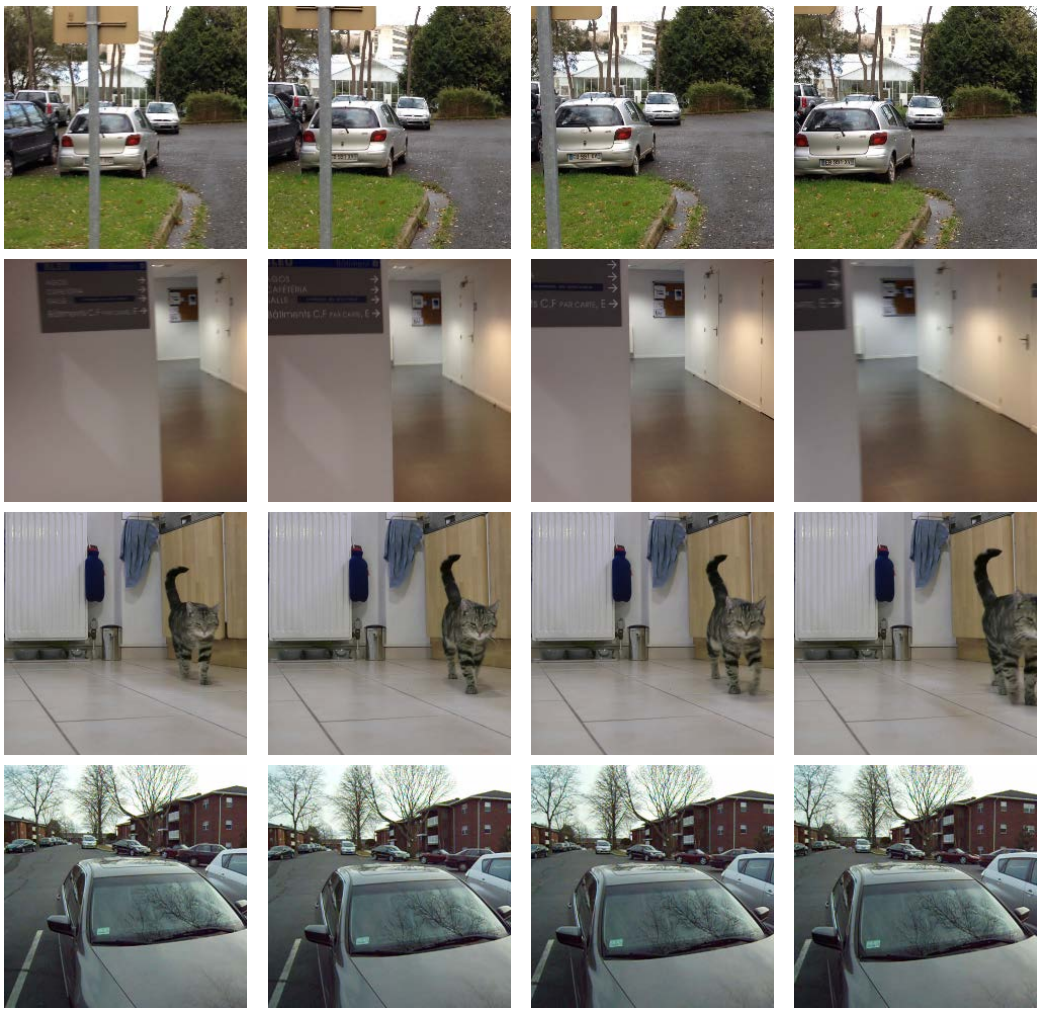


FIGURE 3.2 – Four samples of the real dataset. The first two examples involve non salient frames. The last two ones comprise dynamically salient frames (respectively, including a moving cat, and a (small) moving car behind the parked cars). Four consecutive frames are displayed for each example.

set of 13 videos and a test set of 37 videos, for a total of 3451 labelled frames. The three sets contain approximately the same amount of salient and non salient frames. During training, data augmentation with resizing, cropping, temporal inversion and flipping around a vertical axis, was applied. The batches used to train our networks are built so that salient and non salient frames are correctly balanced. Samples frames are displayed in Figure 3.2.

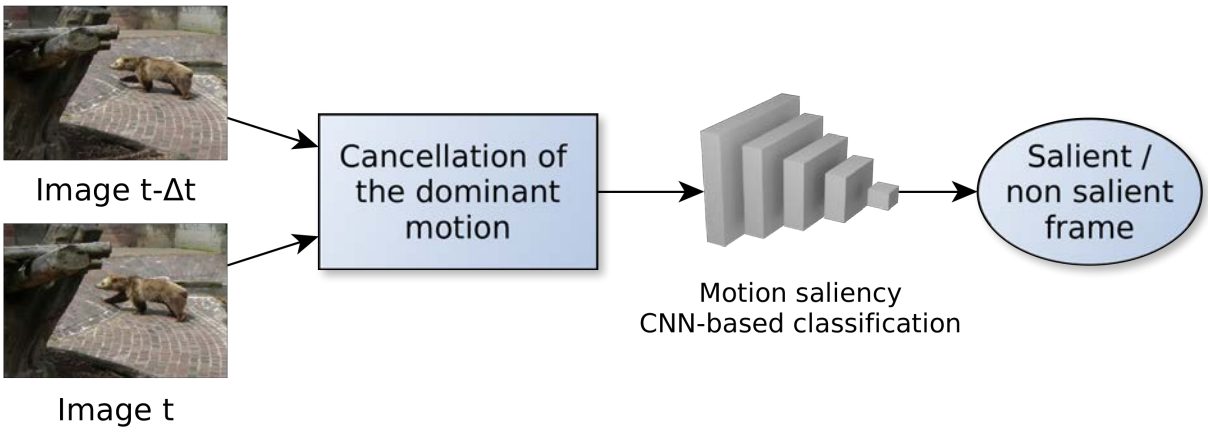


FIGURE 3.3 – Our overall motion saliency detection framework.

	Motion compensation with image warping	Motion compensation with the residual flow
Motion estimation with Motion2D	*WS-Motion2D*	*RFS-Motion2D*
Motion estimation with DeepDOM	*WS-DeepDOM*	*RFS-DeepDOM*

TABLE 3.1 – Four variants of our method, depending on the algorithm estimating the dominant motion, and depending on the way the dominant motion is compensated.

3.3 Temporal motion saliency detection framework

Our overall approach is summarized in Figure 3.3. Its core module is the frame classification, which will be achieved with a convolutional neural network (Section 3.3.1). We have investigated two ways to implement the global motion compensation : image warping (Section 3.3.2), and residual flow (Section 3.3.3).

We assume that the dominant motion (or global motion) can be represented by a single affine motion model. We have considered two parametric dominant motion estimation algorithms, Motion2D and DeepDOM, described in Section 3.3.4. Table 3.1 summarises the four variants of our method depending on the dominant motion estimation and compensation approaches.

In addition, we have designed three baselines for comparison purpose, since there are no existing methods available for temporal motion saliency detection. The first two baselines are merely based on a thresholding step, either in the RGB domain for the image warping variant, or the optical flow domain for the residual flow one (Section 3.3.5.1). These essentially allow us to assess the difficulty of the problem. The third baseline is defined in Section 3.3.5.2. It leverages the two-stream network introduced by [SZ14].

3.3.1 Frame classification based on motion saliency

Due to the clear superiority of CNN-based methods in any image classification problem to date, we adopt a CNN framework for solving the frame classification attached to the temporal detection of motion saliency. The input will be formatted as 240x240 frames, and specified more specifically for each method of temporal motion saliency detection. We have defined the structure of the CNN for classification (Figure 3.4) from preliminary experiments on the synthetic dataset. We conclude from these experiments that a too deep network led to overfitting. Therefore, only three convolutional layers are kept, with 7x7 kernels to have large enough receptive fields. The convolutional layers involve respectively 64, 96 and 128 features maps, with a stride of 2. This architecture also comprises max pooling with a stride of 2, batch normalization, the ReLU non-linearity and dropout.

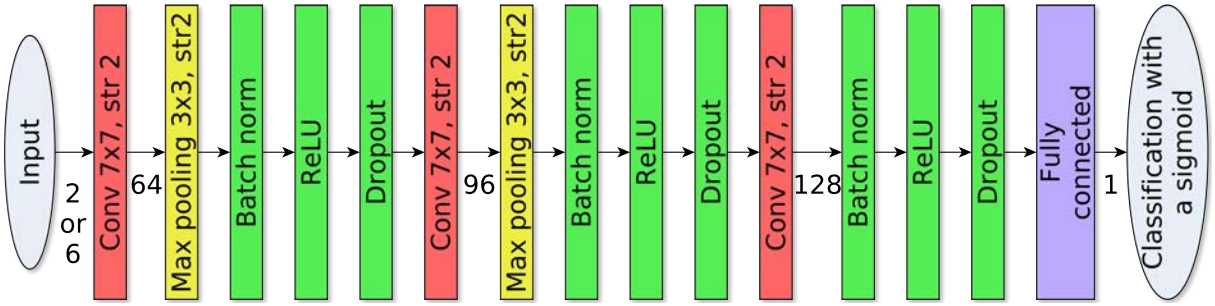


FIGURE 3.4 – CNN for motion saliency classification ('str' stands for stride). The number of channels are mentioned at the relevant places. The input is either the 2-channel residual flow, or the 6-channel concatenation of the two RGB images, or the two-channel optical flow. The final output is the probability of motion saliency, computed with a sigmoid.

Our objective is to build a flexible framework. As a consequence, we keep the same architecture for the classification in all the motion saliency detection methods, whatever the input data. With this specification, we can also draw a fair comparison of the methods. The CNN architecture was trained for each method with the cross-entropy loss.

The network delivers the probability of the frame to be dynamically salient. A frame is then classified as dynamically salient if this probability is greater than 0.5, and non salient otherwise. Notwithstanding, we are not facing a threshold setting issue. Comparing to 0.5 is equivalent to taking the maximum of the two probabilities, when their sum is equal to one. We first designed a network with two output yielding probabilities for the dynamically salient and non-salient classes respectively. We experimentally noticed that the sum of the two probabilities was very close to one, without formally imposing this constraint in the network. More precisely, the mean of the absolute difference between 1 and the sum, on the overall test set, was $3.7\,10^{-5}$, and its maximum value amounts to $4.7\,10^{-4}$, knowing that the test set is well balanced between the two classes. Then, for the sake of simplicity and efficiency, we built a network with one single

output.

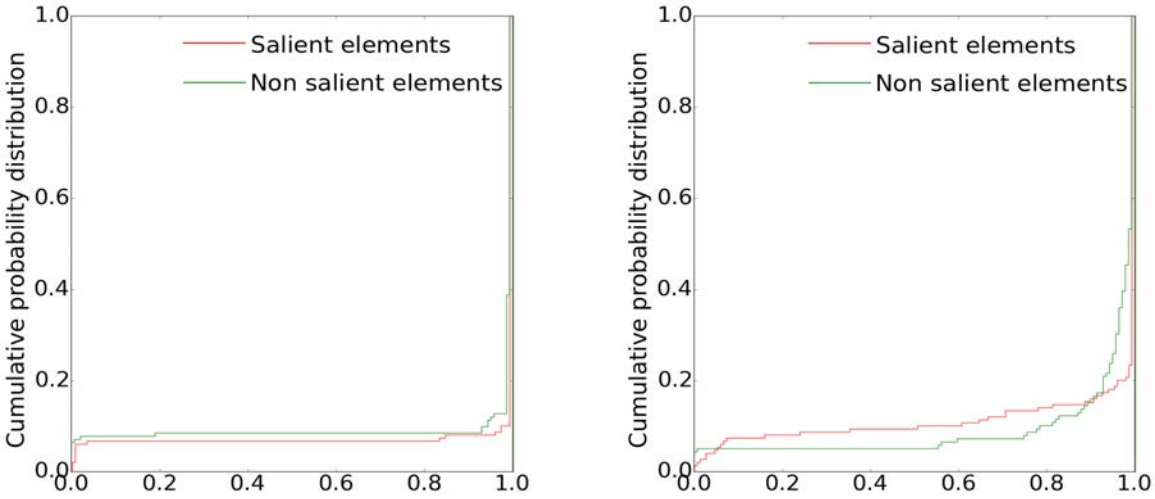


FIGURE 3.5 – Empirical cumulative distribution function for the probability of saliency for salient images (in red), and for the probability of non saliency for non salient images (in green), provided by the method WS-Motion2D (this method will be presented in more detail in section 3.3.2). The plot on the left corresponds to the network trained on synthetic data. The plot on the right corresponds to the network pre-trained on synthetic data and fine-tuned on real data. In both cases, the real validation set has been used to compute the cumulative distribution functions. The optimal time step has been chosen for each network (this aspect will be detailed further in Section 3.4.3).

We plot in Figure 3.5 the empirical cumulative distribution function of the probability of saliency for salient images, and of the probability of non saliency for non salient images. Ideally, the right class is predicted with high confidence : the curves should exhibit values as close to zero as possible for small values, and there should be a peak at 1. Experimentally, the behaviour observed is not that different : there is indeed a high peak close to 1, with also a smaller peak close to 0 (corresponding to a minority of wrong predictions), and the curve is flat in between. This means that the network tends to make predictions with high confidence in all cases. All this confirms that the decision threshold of 0.5 is adapted.

We now present in Sections 3.3.2 and 3.3.3 the two ways compensating the main motion. They lead to different inputs for the classification network presented in Figure 3.4.

3.3.2 Motion saliency detection based on image warping

The first way to cancel the dominant motion is to warp the second image of the pair onto the first one. This is done by displacing each pixel position with the estimated motion, and by

searching in the second image for the color at the corresponding location. Bilinear interpolation is used for non integer positions. Also, the image is zero-padded to account for pixels which would correspond to location outside of the image boundaries.

The method will then rely on the CNN to extract relevant features from the aligned frames to infer whether motion saliency is present or not, i.e., the frames are partly or fully aligned. The classification CNN we use is the network introduced in Section 3.3.1. The two colour images, which are the first input image and the second warped image, are concatenated into a 6-channel input.

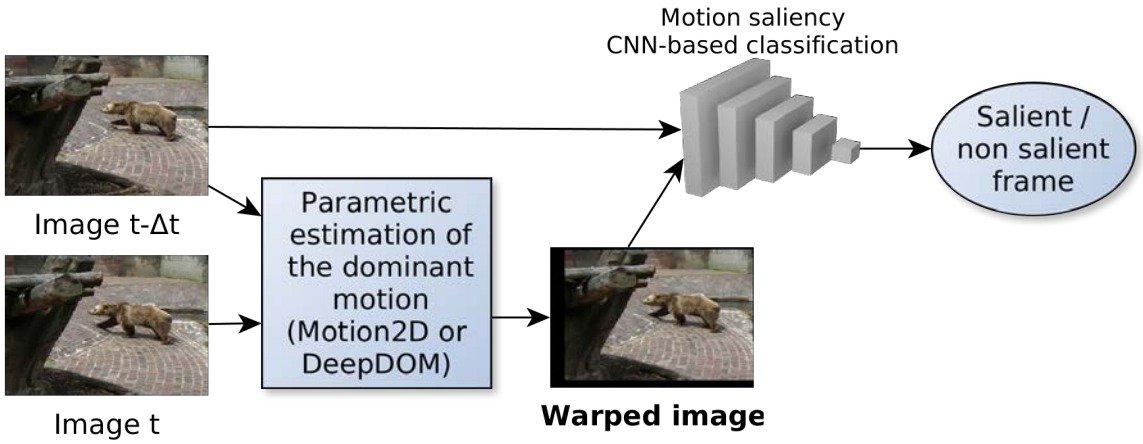


FIGURE 3.6 – Motion saliency detection based on image warping.

We consider the two algorithms Motion2D and DeepDOM for motion saliency estimation. This gives two variants for our methods, which will be called respectively WS-Motion2D or WS-DeepDOM. WS is standing for Warping-based Saliency. The network is expected to compare the warped frames and to find motion saliency by implicitly searching for misaligned parts due to their independent motion.

3.3.3 Motion saliency detection based on the residual flow

Our second motion saliency detection method leverages the fact that motion saliency is first and foremost related to motion. Instead of using colour frames as input, we directly exploit information related to motion. This allows us to be explicitly agnostic on appearance. We compute the residual optical flow, by subtracting the affine flow $\omega_{\hat{\theta}}$, given by the estimated dominant motion model of parameters $\hat{\theta}$, to the computed optical flow field ω :

$$\forall p \in \Omega, \qquad \omega_{res}(p) = \omega(p) - \omega_{\hat{\theta}}(p) \tag{3.1}$$

where Ω is the image grid. The residual flow will serve as input of the CNN classifier of Figure 3.4. The whole workflow of the method is summarized in Figure 3.7. The two components of the residual flow are the two channels of the input of the classifier. A residual flow close to a zero field over the whole image means a non salient frame.

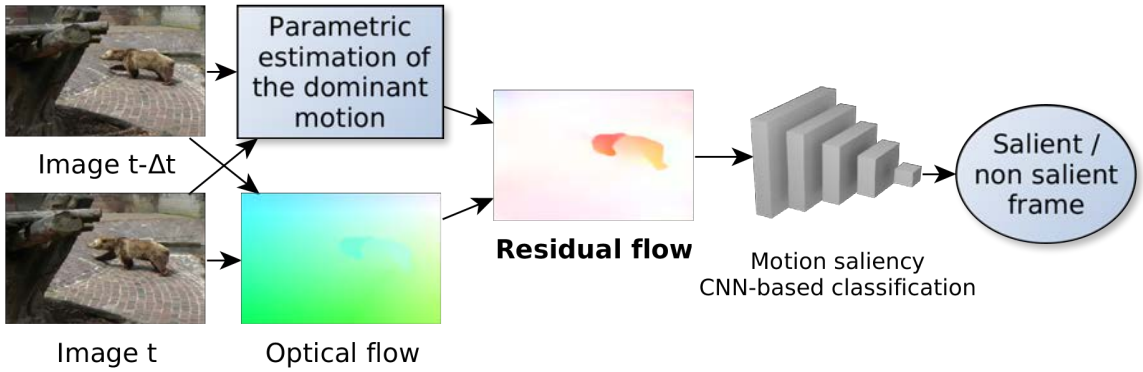


FIGURE 3.7 – Motion saliency classification based on the residual flow.

The optical flow is computed with FlowNet 2.0 [Ilg+17]. We use a trained model provided by the authors as it is, without additional fine-tuning. FlowNet 2.0 was chosen since it is real time, while delivering good performance. The speed of the optical flow algorithm is indeed critical, as optical flow is computed during training for each pair of every batch. Also, expected applications of the motion saliency detection method are likely to require real-time execution. Depending on the motion estimation algorithm used, this method will be denoted RFS-Motion2D or RFS-DeepDOM, with RFS standing for Residual Flow Saliency.

3.3.4 Parametric estimation of the dominant motion

Most of the time, the image motion induced by the camera forms the dominant motion in the frame. This dominant motion corresponds to the apparent motion of static elements in the scene, which usually occupies the main part of the frame. In case of a shallow scene, i.e., depth and orientation variations in the static scene are small compared to the distance to the camera, a unique 2D parametric motion model, such as an affine model, correctly approximates the dominant motion. We adopt this simple but efficient approach. Results reported in Section 3.4 will show that it generalises well even for non shallow scenes in practice. Let us note that for a close-up on a moving object, the dominant motion becomes precisely the motion of this object. The temporal motion saliency detection is still valid, since we are interested in detecting local motion departing from the global one. For every pixel $p \in \Omega$ with $p = (x, y)$, the flow field given

by the parametric motion model can be written as :

$$\omega_\theta(p) = \begin{pmatrix} a_1 + a_2 x + a_3 y \\ a_4 + a_5 x + a_6 y \end{pmatrix} \tag{3.2}$$

where $\theta = (a_1, ..., a_6)$ is the vector of the model parameters.

We used two methods to estimate the affine motion model. We first resorted to the robust multi-resolution algorithm Motion2D [OB95] to estimate the dominant affine motion model between two frames. It is real-time. It has proven its efficiency in many diverse applications that our research group has dealt with, motion detection with a moving camera [OB97], visual servoing [CCB98], cell image stabilisation [Oze+13], crowd motion analysis [PRBB17] to name a few.

As an alternative, we have applied the network proposed in [RAS17], initially designed to estimate geometric transformations between two images, eventually acquired from very distant viewpoints. The objects to match may even be different instances of the same object class. In our case, the two input images are far closer and the displacements are smaller, but the frames may involve outliers, i.e., independently moving objects. The architecture is summarized in Figure 3.8. It was trained, separately from the temporal saliency detection network, with the synthetic dataset described in Section 3.2.1. We applied the architecture of [RAS17] with no modification. In this architecture, the convolutional network before the correlation corresponds to the VGG-16 network [SZ15], cropped at the pool4 layer before ReLU and followed by per-feature L2-normalisation. The convolutional neural network after the correlation operation is composed of two successive convolutional layers with respectively 7x7 and 5x5 kernels, that are followed by a fully connected layer, as illustrated in Figure 3.9. This network is called DeepDOM, for Deep Dominant Motion estimation. The total number of parameters of this network is $16\,891\,136$. For learning, we will consider batches of 25 elements. The loss function used to train this network is similar to the one used in [RAS17]. A grid $\mathcal{G}$ with nodes q is deformed by the estimated global motion of parameters θ and by the ground truth θ_{GT}. Note that the grid is sparse (21x21 nodes) compared to the image (240x240 pixels). The loss function $\epsilon(\theta)$ compares the two sets of grid displacements as follows :

$$\epsilon(\theta) = \frac{1}{|\mathcal{G}|} \sum_{q \in \mathcal{G}} ||\omega_{\theta_{GT}}(q) - \omega_\theta(q)||_2^2. \tag{3.3}$$

3.3.5 Baselines and ablation study

In this section, we present three baselines methods, from which we will derive an ablation study.

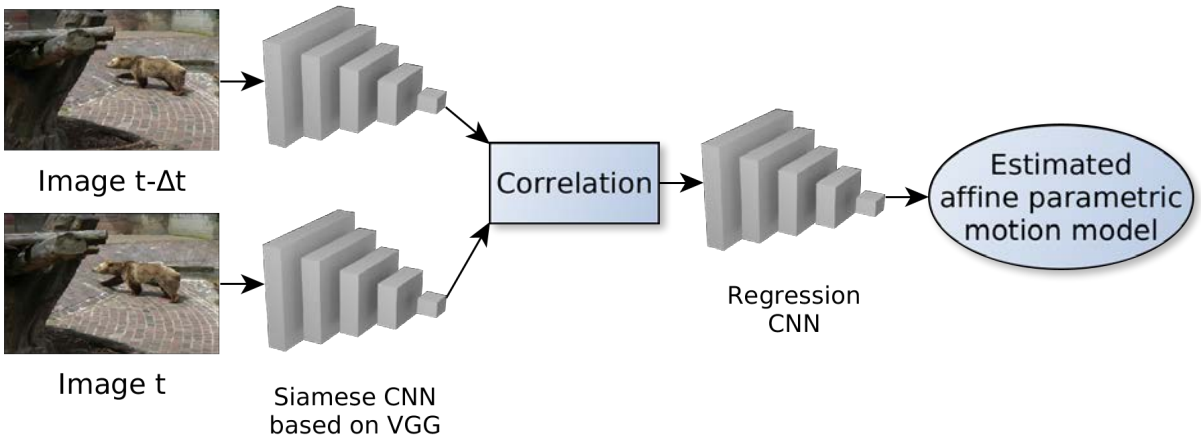


FIGURE 3.8 – DeepDOM network, leveraging the one from [RAS17], for the parametric estimation of the dominant motion.

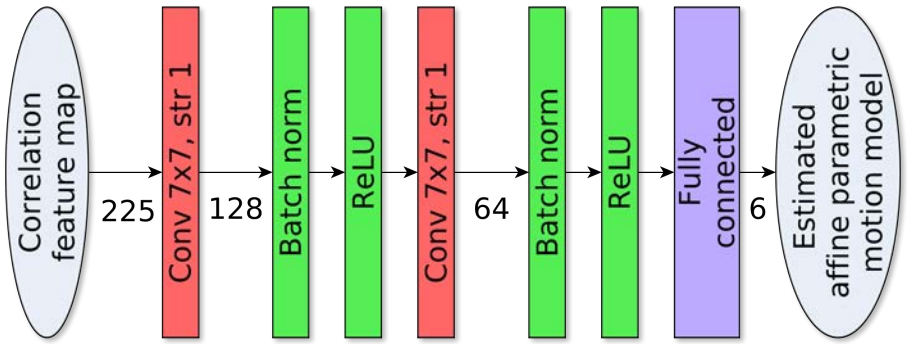


FIGURE 3.9 – Regression part of the DeepDOM network, which leverages the one from [RAS17].

3.3.5.1 Basic baselines

As a starting point for performance comparison, we have defined two simple baselines.

The first one relies on the difference of warped images, similarly to what is presented in Section 3.3.2. The following function Ψ is computed :

$$\Psi(t) = \frac{1}{|\Omega|} \sum_{p \in \Omega} |I(p + \omega_{\hat{\theta}}(p), t+1) - I(p,t)| \tag{3.4}$$

where $\hat{\theta}$ denotes the estimated parameters of the dominant motion model, $\omega_{\hat{\theta}}$ represents the estimated dominant affine flow, I is the image intensity, Ω is the image grid. Motion2D is used to estimate the dominant motion model. Then we compare $\Psi(t)$ to a threshold to decide whether the frame is dynamically salient ($\Psi(t) \geq \lambda$), or not ($\Psi(t) < \lambda$).

In an ideal situation, the warped image would correspond exactly to the first image of the pair, except if there are salient elements that do not undergo the main motion. We have to set the threshold λ. The empirical distribution of function Ψ is computed on the validation set of the real dataset. Then, we fit an exponential distribution to this empirical distribution. We use a p-value test with a probability of false alarm of 5% to determine the threshold λ.

The second simple baseline relies on the residual flow. We compute the following function to decide whether the frame t is dynamically salient :

$$\Phi(t) = \frac{1}{|\Omega|} \sum_{p \in \Omega} \left\| \omega(p,t) - \omega_{\hat{\theta}(t)}(p) \right\|_2 \tag{3.5}$$

where ω is the computed optical flow, $\omega_{\hat{\theta}}$ is the flow field corresponding to the dominant motion model of estimated parameters $\hat{\theta}$, and Ω is the image grid. Again Motion2D is used to estimate the dominant motion, and FlowNet 2.0 to compute the optical flow.

$\Phi(t)$ characterizes the amount of residual motion in the frame t after cancelling the global motion. We threshold $\Phi(t)$ to classify the frame t as dynamically salient or not. To correctly set the threshold value λ, we proceed as for the first baseline. We compute the empirical distribution of $\Phi(t)$ on the frames of the validation set of the real video dataset. Then, an exponential distribution is fit to it. A p-value test with a probability of false alarm of 5% is applied to set the threshold λ.

These two baselines can be interpreted as an ablation study of our method. More specifically, their input is the same as the WS and RFS variants respectively. The difference is that the classification network of our whole framework is replaced by a simple decision criterion. Analysing their performances will show the contribution of our deep classification network.

3.3.5.2 Two-stream network as a baseline

Due to the lack of existing methods to compare with, we designed a third more sophisticated baseline. It is a direct extension of the well-known two-stream network introduced in [SZ14]. The two-stream network involves two CNNs in parallel, one for the spatial stream, and one for the temporal stream. The two-stream network provided convincing results for the action recognition task. It was also exploited in other works, such as in [TBD18] for dynamic texture synthesis and for the prediction of eye-fixation maps in [Bak+18]. We can expect that it is a good candidate for temporal motion saliency detection as well. However, we need to adapt it for this new task. We have to produce a prediction for each frame of a video, and not one prediction for the whole video. Regarding the spatial stream, we concatenate the two considered frames of the video at times $t - \Delta t$ and t. Accordingly, as input of the temporal stream, we take the optical flow field computed between the two frames only. In this version of the two-stream network, we still use the classification CNN of Figure 3.4 for both streams, and FlowNet 2.0 to compute the optical flow.

In addition, we will run each stream separately to provide a complementary ablation study of our own method. Indeed, the spatial and temporal streams are related to the WS and RFS variants of our method respectively. However, the key difference is that they do not include the dominant motion compensation step (no image warping or no residual flow computation respectively). The two-stream network is trained the same way as our main method, with the cross-entropy loss. For the ablation study, the same weights than for the whole two-stream network are used (the streams are not retrained separately). The resulting network is summarized in Figure 3.10.

3.4 Experimental results

3.4.1 Experimental setting

We use the Caffe library [Jia+14] to implement the networks presented in Section 3.3. The optimization was achieved with the Adam method [KB14] with the parameters proposed by the authors for all the networks. The runtime for the processing of a batch during the learning stage (prediction and back-propagation), with a GPU Tesla M40 and a 2.9 GHz processor is respectively of 1.4 sec, 1.8 sec, 0.7 sec and 1.2 sec for WS-DeepDOM, WS-Motion2D, RFS-DeepDOM and RFS-Motion2D. The batch size consists of 32, 32, 8 and 12 elements respectively. The last two batch sizes are lower due to a higher GPU memory requirement of the networks, in particular for the optical flow computation with FlowNet 2.0. The total number of parameters of these networks is $17\,817\,729$ for WS-DeepDOM, $926\,656$ for WS-Motion2D, $180\,299\,060$ for RFS-DeepDOM, and $163\,407\,924$ for RFS-Motion2D. In the test stage, the prediction for one frame is

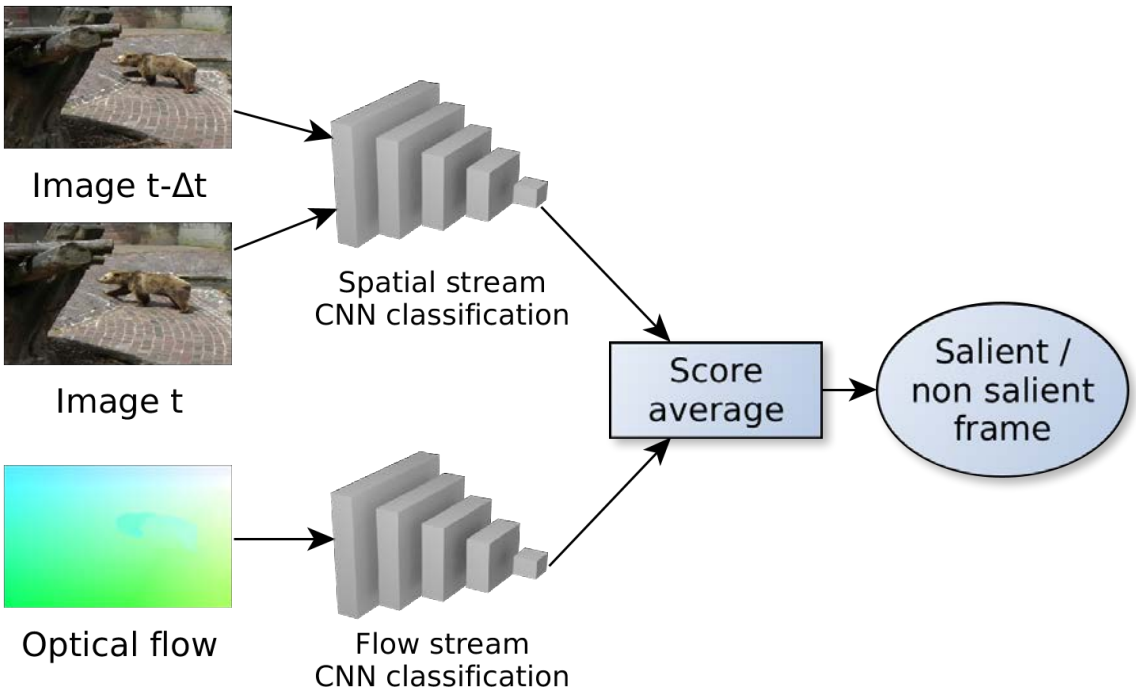


FIGURE 3.10 – Two-stream network for temporal motion saliency detection. The spatial and temporal streams follow the same architecture, as the one detailed in Figure 3.4. The input of the spatial stream is the 6-channel concatenation of the RGB images at times $t - \Delta t$ and t. The input of the temporal stream is the two-component optical flow map. The final saliency score is the average between the spatial and temporal saliency scores.

performed in respectively 20.0, 15.2, 10.4 and 9.5 fps.

3.4.2 Comparison of the dominant motion estimation methods

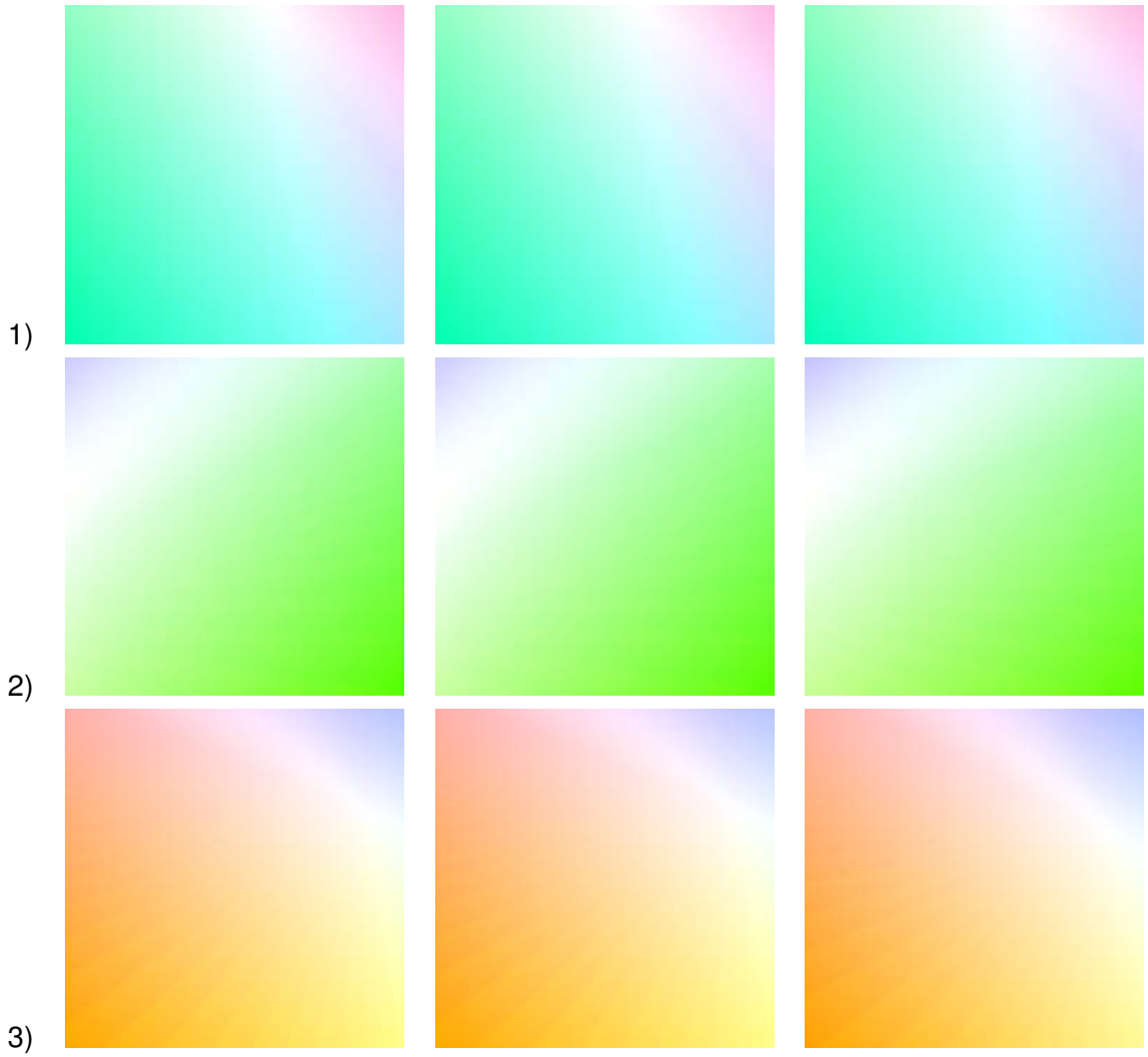
The accuracy of the dominant motion estimation methods is likely to play an important role in the performance of the temporal motion saliency detection methods. We evaluated them on the synthetic dataset, which contains 2000 elements for which the ground truth is available by construction. For this evaluation, we proceed as for the loss used to train DeepDOM (Equation 3.3), by estimating the reconstruction error on a grid for each element. With DeepDOM, we obtained a mean error for the estimation of the dominant motion of 0.20 pixels with a standard deviation of 0.08. With Motion2D, the mean error is of 0.03 pixels and the standard deviation is of 0.41. For the synthetic dataset, Motion2D provides an estimation of the dominant motion closer to the ground truth. Yet, the accuracy obtained by DeepDOM is good, and the standard deviation of its error is lower. As illustrated in Figure 3.11, the flows estimated by these two methods are visually close most of the time.

3.4.3 Choice of the time step

Another important aspect is the choice of the time step Δt between the two frames for the prediction stage. This discussion assumes standard frame rates around 25 to 30 fps. A time step of 1 means that we consider two successive frames of the video, a time step of 2, frames at $t-2$ and t, etc. We can expect that more distant frames will make the highlight of independently moving objects with small motion magnitude easier. On the other hand, the parametric estimation of the dominant motion is supposed to be more precise on temporally closer frames. Table 3.2 collects the accuracy of the RFS-Motion2D method for different time steps. To find the best trade-off, time step was set to 1 during training, and the validation set with real videos is used to select the best time step for every method. Each method has thus the most appropriate time step for testing. It will also make the comparison fairer.

Time step Δt	1	2	3	4	5	6	7	8	9
Accuracy	84.9	89	88.3	88.9	**89.6**	88.9	88.9	89.2	88.5

TABLE 3.2 – Rates of correct classification in percentage for RFS-Motion2D on the real validation set for growing time steps Δt from 1 to 9.



Method	Sample	a_1	a_4	a_2	a_3	a_5	a_6
Ground truth	1)	-6.31	1.18	0.044	-0.043	-0.030	0.005
Motion2D	1)	-6.31	1.18	0.044	-0.043	-0.030	0.005
DeepDOM	1)	-6.28	1.01	0.041	-0.043	-0.028	0.002
Ground truth	2)	-1.97	5.13	-0.007	0.004	0.033	0.039
Motion2D	2)	-1.97	5.12	-0.007	0.004	0.033	0.039
DeepDOM	2)	-1.77	4.78	-0.007	0.006	0.030	0.040
Ground truth	3)	3.95	3.05	-0.030	0.024	-0.014	0.036
Motion2D	3)	3.95	3.05	-0.030	0.024	-0.014	0.036
DeepDOM	3)	3.98	2.85	-0.033	0.026	-0.017	0.035

FIGURE 3.11 – Visualisation of the ground truth affine dominant motion (left image), motion field estimated with Motion2D (middle image) and with DeepDOM (right image) for three samples. The table below gives the numerical value of the parameters in each case. The motion model is described by equation 3.2.

3.4.4 Experimental evaluation and comparison

3.4.4.1 Evaluation on the synthetic dataset

All the learning-based motion saliency detection methods were trained first on the synthetic database introduced in Section 3.2.1. The evaluation on the synthetic validation set showed that for all the methods, the accuracy was higher than 98%. Such a good performance can be explained by the fact that the synthetic dataset is ideal, as it displays strictly affine motions.

3.4.4.2 Evaluation on the real video dataset

Table 3.3 contains the comparative results. First of all, the two simple baselines relying on criteria (3.4) and (3.5) yield an accuracy of only 50.2% and 54.4% respectively. It demonstrates that the problem is not trivial, which justifies the use of a CNN-based framework for the classification. The two-stream network has an accuracy of 80.9%, which is better, but leaves room for improvement. Let us note that the spatial and temporal streams, when used separately to make the prediction, do not provide equivalent results. The temporal stream performs better than the spatial stream by a large margin. This confirms that for the temporal motion saliency detection task, explicit motion information is the key input. In contrast to the action classification task investigated in [SZ14], no improvement was obtained by combining the two streams. By comparing the spatial and temporal streams to our WS and RFS variants respectively, we observe that our method consistently obtains better performance. This demonstrates the contribution of the motion compensation step in both cases.

If we now compare the variants of our method in Table 3.3, we see that the best method is RFS-Motion2D, which is based on the residual flow. It reaches an overall accuracy of 87.5%. Globally, we draw the same conclusion than for the two-stream baseline, that is, for a given dominant motion estimation algorithm, methods which take flow as input perform better that methods that take images as input.

Samples of results are displayed in Figure 3.12 for visual assessment. In the processed videos, the static scene is not always shallow, and moving objects may be not easily visible. However, the classification results remain convincing as illustrated in the top rows of Figure 3.12. Difficult cases involving static objects in the foreground as trees or walls, dynamic textures as a flowing river, or camouflaged moving objects as the snail, are yet correctly classified. The bottom row of Figure 3.12 contains failure cases for frames involving wind in the bushes, or small moving objects (respectively car, dog, scorpion) partially hidden.

Method	Time step	Overall	Salient frames	Non salient frames
Baseline Ψ	4	50,2	87,0	8,2
Baseline Φ	4	54.4	64.6	42.8
Two-stream network	1	80.9	64.3	99.7
Temporal stream only	2	83.2	71.4	96.7
Spatial stream only	3	72.6	50.4	98.0
WS-Motion2D	6	<u>85.2</u>	78.4	93.0
WS-DeepDOM	2	76.5	59.4	96.2
RFS-Motion2D	5	**87.5**	79.7	96.4
RFS-DeepDOM	3	84.6	73.0	98.0

TABLE 3.3 – Rates of correct classification on the test set with real videos. Best performance in bold, second best underlined.

Method	Time step	Overall	Salient frames	Non salient frames
WS-Motion2D affine	6	85.2	78.4	93.0
WS-Motion2D quadratic	6	85.2	77.9	93.5

TABLE 3.4 – Rates of correct classification on the test set with real videos for WS-Motion2D, depending on the choice of the parametric motion model.

Method	Overall	Salient frames	Non salient frames
WS-Motion2D	**80.9**	76.8	85.7
WS-DeepDOM	67.3	62.3	73.1
RFS-Motion2D	76.0	62.2	91.8
RFS-DeepDOM	69.7	44.1	99.0

TABLE 3.5 – Rates of correct classification on the test set with real videos, where methods are trained only on the *synthetic* dataset.

3.4.5 Impact of the quality of the dominant motion estimation

Using DeepDOM did not improve the results compared to the classical method Motion2D. This suggests that Motion2D still outperforms DeepDOM for real images. Methods WS and RFS are not equally affected by a lower accuracy of the dominant motion estimation method on real videos. Table 3.3 shows that the performance decrease is 2.9% for RFS and 8.7% for WS with the DeepDOM variant. This suggests that RFS is more robust to a less precise motion estimation method. Intuitively, WS really needs a correct cancellation of the camera motion when comparing registered pixels at the same location to detect motion saliency. In contrast, optical flow contains information directly exploitable, and the camera motion compensation acts rather as a “denoising” step.

3.4.5.1 Choice of the parametric motion model

All the reported experiments involve an affine motion model. To evaluate how the chosen motion model impacts the classification performance, we ran WS-Motion2D with two different motion models : the affine one and the 8-parameter quadratic one. The latter consists of a polynomial of degree 2 for the two components of the velocity vector for a total of 8 free parameters. It accounts for the 2D projected motion of a 3D rigid motion of a planar scene, similarly to the homography for geometric transformation. The WS-Motion2D method has been chosen for this comparative evaluation, since it is more affected by the quality of the global motion estimation as mentioned above. The classification CNN is not trained again. Results reported in Table 3.4 show that modifying the motion model at test time has almost no impact on the performance.

3.4.6 Impact of fine-tuning

We also want to evaluate the impact of the fine-tuning stage. Table 3.5 contains the rates of correct classification for the methods trained only on the synthetic dataset. By comparing Tables 3.3 and 3.5, we notice that further training on real videos always improves the performance. WS-Motion2D has already good performance with training on synthetic data only, but the other methods really benefit from the fine-tuning on real videos.

3.4.7 Temporal evaluation

The temporal behaviour of the four variants is illustrated in Figure 3.13 with timeline plots. The 12 real video clips respectively depict moving camel, cars (twice), cat, tractor, giraffe and scorpion for the seven dynamically salient clips, and field, river, countryside, campus and indoors for the five non salient ones. These videos are all represented in Figure 3.12. All the methods yield more stable results on non salient videos than on dynamically salient ones. The

seventh clip in the row is a very difficult example (the scorpion one) of the Camouflaged Animals dataset. RFS methods are able to partly detect motion saliency in this video, whereas WS methods fail, demonstrating that motion information is the key clue and appearance may be useless.

Results obtained with RFS-Motion2D for a subset of the test set, and concatenated in a single video, can be found at the following link : `https://youtu.be/sQjSmBPAkaU`. The colour of the frame border, respectively orange or blue, designates the prediction, respectively dynamic saliency or non saliency. The green (respectively red) square at the bottom right indicates that the prediction is correct (respectively wrong). The video shows that non saliency is correctly predicted even when the shallow scene assumption is not valid, with for instance walls, trees or pillars in the foreground. The video includes the camouflaged snail case which is successfully handled.

3.4.8 Additional experiments

We applied RFS-Motion2D to the DAVIS2016 dataset [Per+16], which includes 50 videos with 3455 annotated frames. It was initially built for the video object segmentation task. The objects of interest are foreground moving objects. All the frames should be labelled as dynamically salient. We run RFS-Motion2D without any fine-tuning on the DAVIS2016 dataset. We obtained an overall correct classification rate of 93.3%, which is significantly better than the one obtained for our real dataset (79.7% for salient frames, see Table 3.3). Moving objects in our dataset usually exhibit a smaller motion magnitude than those of the DAVIS2016 dataset. Our dataset also includes the challenging Camouflaged Animals samples.

In addition, we conducted an experiment regarding the claim made in Section 3.1 about the applicability of existing saliency map estimation methods to saliency detection. We applied the method of [WSS18] on the non-salient videos, using the code made available by the authors. As expected, the method supplied non-empty motion saliency maps, sometimes involving large non-salient areas. To be fair enough, we added a decision step for an effective classification. For this, we considered the saliency map delivered by the final sigmoid layer of their method, which has values in [0,1]. These values represent the probability of saliency at each pixel. Then, we consider that a frame is salient if at least 100 pixels have a saliency value of more than 0.5, for images of dimension 240x240. With this procedure, we got a correct classification rate of non-salient frames of 58%. In contrast, our method reached a rate of 98% (Table 3.3). This shows that existing methods are not able to solve the problem we have addressed.

3.5 Conclusion

We have formulated the problem of temporal motion saliency detection in videos. We proposed two methods involving camera motion compensation and CNN-based classification. They were favourably compared to a baseline exploiting the two-stream network. A synthetic dataset has been constructed and an existing real dataset has been extended. The best method (RFS-Motion2D) reaches an overall accuracy of 87.5% on our real dataset, and even 93.3% on the DAVIS2016 dataset. It takes advantage of an explicit motion information given by the residual flow.

A challenge for temporal motion saliency detection methods is to properly handle static objects close to the camera, such as trees or pillars, which can have a strong apparent motion. Our method learned to properly classify such samples most of the time, as shown by the high accuracy on non salient frames. On the other hand, our method is also more careful when predicting saliency and some salient elements are missed. A perspective would then be to search for a way to better separate apparent and real motion, to make the saliency estimation easier.

FIGURE 3.12 – Classification examples for RFS-Motion2D. The four top left rows and top right rows show eight frames properly classified as respectively non dynamically salient and dynamically salient. The bottom row displays four failure cases. The first frame is wrongly classified as dynamically salient due to wind in the bushes ; the red circle surrounds the salient moving object in the three other frames that are wrongly classified as non salient.

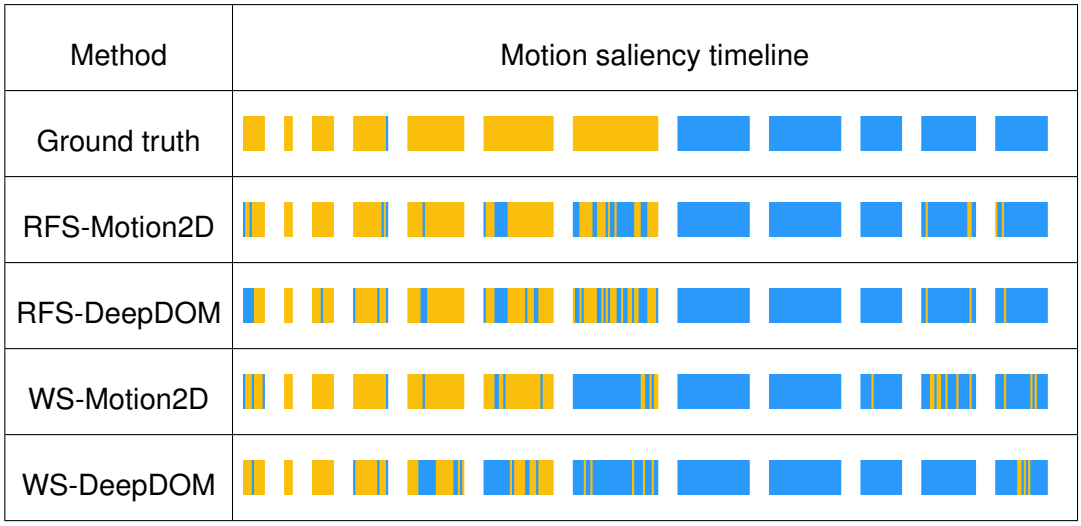


FIGURE 3.13 – Comparative timelines of the frame classification on 12 real videos of different lengths (orange stands for dynamically salient, blue for non-salient).

CHAPTER 4

MOTION SALIENCY MAPS

This chapter addresses the problem of motion saliency in videos, that is, identifying regions that undergo motion departing from its context. We propose a new unsupervised paradigm to compute motion saliency maps. The key ingredient is the flow inpainting stage. Candidate regions are determined from the optical flow boundaries. The residual flow in these regions is given by the difference between the optical flow and the flow inpainted from the surrounding areas. It provides the cue for motion saliency.

4.1 Introduction

Estimating motion saliency maps consists in locating in each frame of a video the saliency induced by motion. More precisely, in each frame of the video, regions whose motion significantly departs from the surrounding motion will be considered to be salient. Estimating motion saliency can provide useful information for various applications, such as mobile robotics or autonomous vehicles [DRT18 ; RW20], alert raising for video surveillance [SRB12 ; DDM19], or the identification of temporal segments of interest for video summary [LPC09 ; SCB12].

Related works concerning saliency in videos were reviewed in Section 2.5 of Chapter 2. Here, let us simply summarise the general trends followed by the main methods of the literature. First, most existing works talk about *spatio-temporal* saliency or *dynamic* saliency or *video* saliency. The notion of saliency associated to these terms takes into account not only motion but also appearance. More specifically, the primary goal of these methods for saliency estimation is either to predict the visual attention of human observers, or to highlight the moving object of interest in the foreground of the video. We note that in the first case, it is well-known that moving objects tend to attract the attention of observers. In particular, a strong motion in the periphery of the field of view will act as a signal which attracts the attention of the observer. In the second case, motion generally provides a reliable indication regarding the presence of any object of interest.

In contrast to most existing approaches, we will deal with *motion saliency* which, by definition and as opposed to *appearance saliency*, involves only motion information. We do not exploit appearance cues and we do not make any specific hypothesis related to this aspect.

Our method is on one hand more focused, as it deals exclusively with motion, but on the other hand it is also more general. Indeed, in quite common situations, saliency may be due to motion exclusively. This is for instance the case for anomaly detection in crowds [PRBB17]. An individual may follow a different motion compared to the surrounding crowd, and similar situations can occur for an animal in a herd or flock, a car in the traffic, or a cell in a tissue. Notably, appearance may be of limited use for specific imaging modalities, for instance videos taken with an infrared camera or for fluorescence microscopy.

In addition, our method requires no learning stage, be it supervised or unsupervised. Our main contribution is the introduction of *optical flow inpainting* to design an original and effective solution for the problem of motion saliency estimation. Relying on optical flow is natural as it provides dense information about the motion present in the video. In practice, any optical flow method can be used, provided it is precise enough, in particular regarding the preservation of motion boundaries, and provided the computation time it requires remains low. The optical flow can indifferently be estimated by classical methods, for instance by variational approaches, or by methods built on deep neural networks. The latter are now leading. In both cases, we exploit methods without any modification, and we take the optical flow produced as input of our method. Our method then processes this raw optical flow to extract motion saliency, and this processing involves no supervision or learning stage.

The rest of this chapter is organised as follows. Section 4.2 presents our method for the estimation of saliency maps. Section 4.3 reports comparative results with state-of-the-art methods of video saliency. Section 4.4 contains concluding comments.

4.2 Estimation of motion saliency maps

As stated in the introduction, we estimate motion saliency maps in video sequences only from optical flow cues. We expect that the optical flow field will be distinguishable enough in salient regions. We have to compare the flow field in a given area, likely to be a salient moving element, with the flow field that would have been induced in the very same area with the surrounding motion. The former can be computed by any optical flow method. The latter is not directly available, since it is not observed. Yet, it can be predicted by a flow inpainting method. This is precisely the originality of our motion saliency approach. Our method is then two-fold. First, we extract candidate salient regions and compare the inpainted flow to the original optical flow in these regions. The discrepancy between the two flows is then interpreted as an indicator of motion saliency. We additionally include a backward and forward processing to refine the results. Our overall framework is summarised in Figure 4.1.

In Sections 4.2.1, 4.2.2, 4.2.3 and 4.2.4, we will describe the successive steps of our method. This description is accompanied by Figures 4.4 to Figures 4.12 (starting at page 87) to

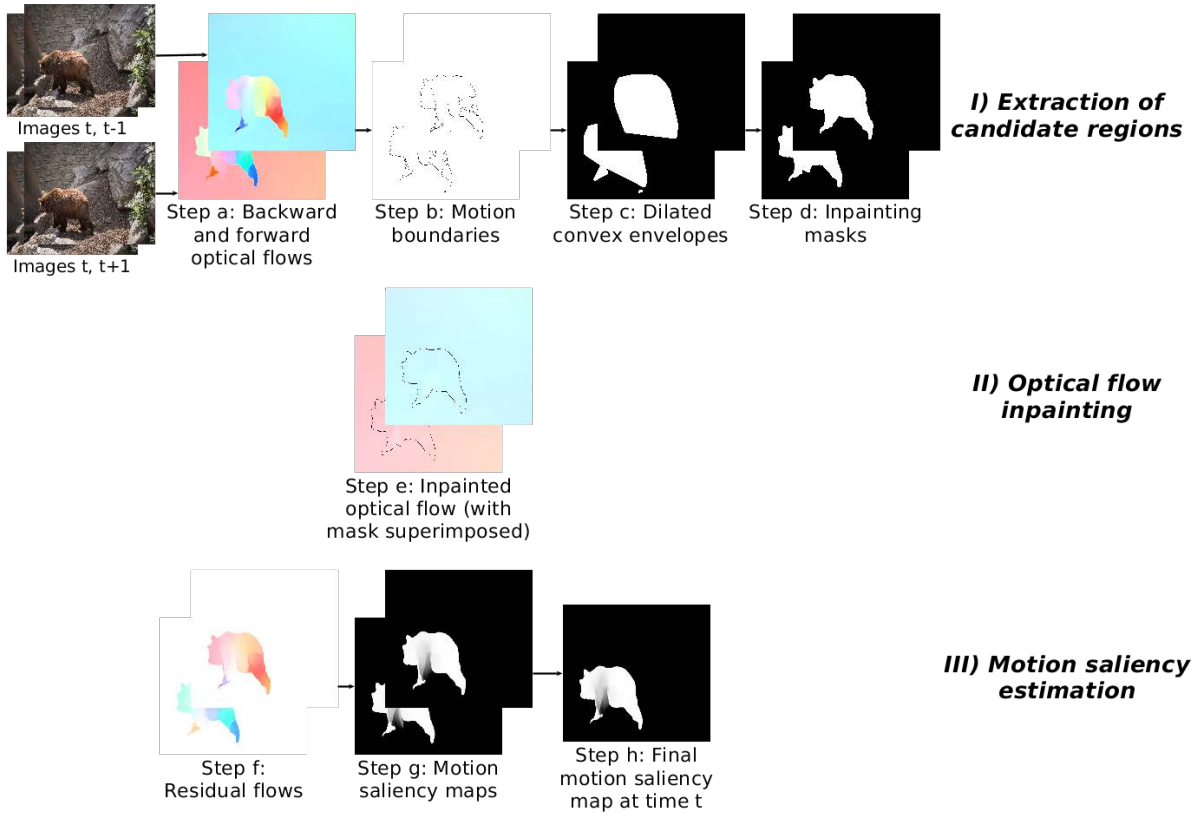


FIGURE 4.1 – Overall framework of our method for motion saliency map estimation with the two backward and forward streams. Steps a) to d) estimate candidate salient regions. Motions boundaries are extracted from the HSV representation of the optical flow. The convex envelopes are the basis on which, after additional refinements, the final candidate regions are obtained in c) and d). In step e), the optical flow in the candidate regions is reconstructed from the external flow. The residual flow, which is the difference between the optical flow and the inpainted flow, is computed in f). It is then converted into the backward and forward saliency maps in g), that are merged into the final saliency map in h).

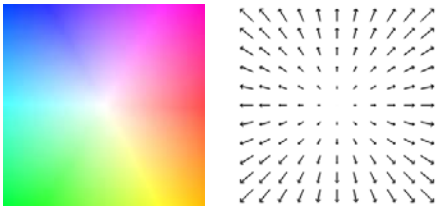


FIGURE 4.2 – Colour code (left) for the corresponding optical flow field (right).

get a better insight of the role of the different steps in different situations. Figure 4.4 displays eight samples of videos that will be processed, and Figures 4.5 to Figures 4.12 illustrate the forward and the backward streams of the workflow. The backward stream is presented on the left and the forward one on the right for each sample. The *cows*, *motorbike*, *soapbox*, *soccerball* and *Lucia* samples correspond to situations in which the method works as expected, the *dog* and *goat* samples correspond to failure cases, and the *Libby* sample illustrates a challenging setting.

4.2.1 Extraction of inpainting masks

First, we have to extract the masks of the regions to inpaint. We will rely on the optical flow field computed over the images. The optical flow extracted with FlowNet 2.0 [Ilg+17] is displayed in Figure 4.5 for several samples. FlowNet 2.0 has been chosen for its low computation time and sharp motion boundaries. We will exploit the discontinuities of the optical flow. Indeed, the silhouette of any salient moving element should correspond to motion boundaries, since its motion should differ from the surrounding motion. The surrounding motion is generally given by the background motion, as can be seen in the examples of the figure. In the sequel, the surrounding motion will also be referred to as the global motion.

For the motion boundary extraction, one could directly apply a threshold on the norm of the gradient of the velocity vectors. This is however likely to produce noisy contours. Instead, we choose to rely on the classical contour extraction method proposed by Canny [Can86]. To do this, we convert the optical flow to its HSV representation, which is taken as input to the Canny algorithm. The HSV representation of the flow is commonly used for visualisation, the hue representing the direction of motion and the saturation its magnitude (see Figure 4.2). Figure 4.6 displays contours we obtained this way. The location of the salient object can be visually inferred in most cases when looking at these contours. Still, at this point, the contours remain curves or sets of curves which are 1-dimensional objects. They do not straightforwardly allow to find the location of the salient elements yet.

Converting the contours to 2-dimensional masks is the goal of the following step. To this end, contours pieces are first organised into connected parts. A connected part is simply defined by the following property : there exists an uninterrupted path between any two points of the connected part that goes only through other points of this same connected part. More precisely,

we consider the 8-connexity, and a given pixels is connected to its 8 neighbours. The convex envelope of each connected part is then computed and dilated with a local kernel. In practice, we take a kernel of size 5x5. Each resulting region mask is given by the corresponding union of overlapping dilated convex envelopes. At this point, let us note that it is not a problem if several close moving objects are covered by a unique mask. Indeed, these moving objects are potentially salient and should be found as such by our method. For example, the *soapbox* sample displays such a configuration, with several moving objects (two people and the vehicle) that undergo a salient motion, as can be seen in Figure 4.4.

By construction, we note that region masks tend to be larger than actual salient areas. This can be seen in Figure 4.7 for most masks. Nevertheless, this is desirable for inpainting, since inpainting must start from global motion information only.

Yet, a too rough mask can affect the accuracy of motion inpainting, especially for the salient areas which are non convex. The masks are then refined by applying the GrabCut algorithm [RKB04] on the HSV representation of the optical flow. To avoid small localisation errors, which would make the inpainting stage reconstruct the flow from salient pixels, a dilatation with a 5x5 kernel is again applied to the mask resulting from the application of GrabCut. As illustrated in Figure 4.8, the final borders of the masks correspond better to those of objects. This is particularly visible for the *soapbox* sample. The *Libby* sample also shows that this step can do more than refining the borders of the mask, by discarding some regions with a flow very similar to the background. This is a limit case of the GrabCut algorithm, which can discard the whole mask if it is very similar to the background.

4.2.2 Optical flow inpainting

At this point, we have a set $\mathcal{M}$ of inpainting masks r in the image domain Ω for each video frame. The issue now is to perform the flow inpainting in these masks from the surrounding motion. Note that the idea of relying on the reconstruction of motion has been proposed in [Mat+06] for shaky video stabilisation or [SDC14] for video completion. The inpainting problem was first popularised for static images leading to image inpainting methods categorised in exemplar-based methods [CPT03] and diffusion-based methods [Ber+00].

For this step, we adopt the floating-point representation of the velocity vectors $\{\boldsymbol{\omega}(p), p \in \Omega\}$ with $\boldsymbol{\omega}(p) \in \mathbb{R}^2$. The two components of the flow vectors will be inpainted separately. We have investigated three inpainting techniques for this stage. The first two are PDE-based methods [BBS01 ; Tel04]. Since the background motion to inpaint is globally smooth, a diffusion-based approach for inpainting is well-suited. The last approach is a parametric method. In the following, we describe each of these methods, as well as how we have applied them for optical flow inpainting.

4.2.2.1 Navier-Stokes inpainting

For the definition of their inpainting method, the authors of [BBS01] take inspiration from fluid dynamics. Their idea is to rely on the Navier-Stokes equation to propagate lines of equal brightness (also called isophotes) continuously into the region to inpaint. More precisely, they consider the following vorticity transport equations :

$$\frac{\partial w}{\partial t} + v \cdot \nabla w = \nu \nabla \cdot (h(|\nabla w|)\nabla w), \tag{4.1}$$

with $w = \Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$ (I is the image intensity, x and y are the image coordinates) and $v = \nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$. The coefficient ν is the coefficient of anisotropic diffusion and h is a function allowing the anisotropic diffusion of w. The image intensity I is retrieved by solving simultaneously equations 4.1 and 4.2 :

$$\Delta I = w, \qquad I|_{\partial r} = I_0, \tag{4.2}$$

with I_0 the intensity outside of the inpainting mask and ∂r the boundary of the inpainting mask.

We rely on the OpenCV [Bra00] implementation of the method of [BBS01]. To apply it to the inpainting of the optical flow, we replace the image intensity I with the horizontal and vertical velocity fields constituting the optical flow.

4.2.2.2 Telea inpainting

The second variant we investigate consists in adapting the image inpainting method based on fast marching [Tel04] to optical flow inpainting. This approach was adopted by [SDC14] for video completion. In [Tel04], the pixels are progressively inpainted with the following formula :

$$I(p) = \frac{\sum_{q \in B_\epsilon(p)} w(p,q)[I(q) + \nabla I(q)(p-q)]}{\sum_{q \in B_\epsilon(p)} w(p,q)}, \tag{4.3}$$

with p the pixel to inpaint, $B_\epsilon(p)$ a neighbourhood of p with known values, and $w(p,q)$ a weighting function. The pixels are inpainted iteratively. The fast marching method is used to find the exact order in which the pixels are inpainted : the pixels closest to the mask boundary are inpainted first, and the greater the distance, the latter a given pixel will be inpainted.

For the method of [Tel04], we rely on the available OpenCV implementation. To apply it to the optical flow, we proceed similarly as for [BBS01], by replacing the image intensity I with the horizontal and vertical velocity fields constituting the optical flow.

4.2.2.3 Parametric inpainting

Finally, we developed a parametric alternative, which is based on different principles compared to the two methods presented above. We assume that the surrounding motion, i.e., the background motion, can be approximated by a single affine motion model on the whole image as follows :

$$\begin{cases} \omega_x(x,y) = a_0 + a_2 \cdot x + a_3 \cdot y \\ \omega_y(x,y) = a_1 + a_4 \cdot x + a_5 \cdot y \end{cases}, \tag{4.4}$$

with x and y the coordinate of a point in the image, ω_x and ω_y the horizontal and vertical components of the optical flow vector, and $(a_0, a_1, a_2, a_3, a_4, a_5)$ the parameters of the motion model.

For the estimation of this affine motion model, we make use of the method Motion2D [OB95]. This method develops a robust approach to ensure a good performance in presence of secondary motion fields. A multiresolution processing is included to get better performances. In this case, the inpainted flow simply corresponds to the flow issued from the estimated affine motion model, given by equation 4.4, over the masks.

With these three approaches for optical flow inpainting, we have three variants of our method. They are respectively named MSI-ns, MSI-fm and MSI-pm. MSI stands for Motion Saliency Inpainting, ns for Navier-Stokes, fm for fast marching, pm for parametric. We note that it is straightforward to replace the inpainting method in our framework by any other of our choice. Examples of other inpainting algorithms not investigated here include the recent methods of [Oli+18] or [Raa+20], specifically designed for optical flow inpainting.

Figure 4.9 shows the inpainted flows for the MSI-ns variant. We can see that in most cases, the inpainted flow is globally smooth and the salient object is no longer visible. This means that the mask was large enough to cover the whole object. Additionally, Figure 4.3 compares the inpainting with the three methods for the cow sample. Qualitatively, the reconstructed flows are smooth, except for the Telea variant which leads to a more granular reconstruction. We also observe the apparition of an artifact close to the center of the inpainting mask with this variant. Still, the reconstructed flow is in all cases different from the original flow for this sample. The more detailed quantitative evaluation of the impact of the inpainting method on the final result will be given in Section 4.3.2.

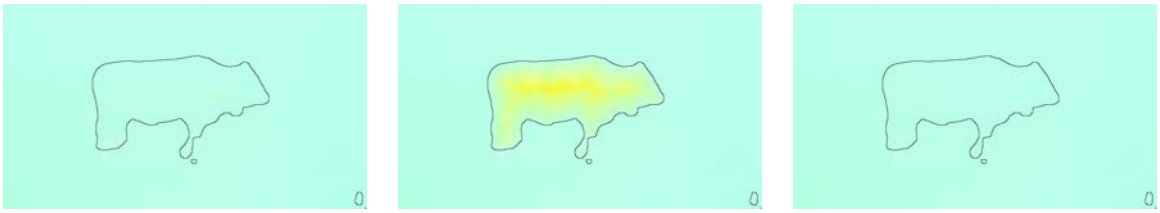


FIGURE 4.3 – Optical flow inpainted with the Navier-Stokes method [BBS01] (left), the Telea method [Tel04] (center) and the parametric method [OB95] (right). The inpainting mask is superimposed to the flow. The reconstructions are globally smooth, except for the Telea variant which is more granular.

4.2.3 Motion saliency map computation

The residual flow $\boldsymbol{\omega_{res}}$, i.e., the difference between the optical flow $\boldsymbol{\omega}$ and the inpainted flow $\boldsymbol{\omega_{inp}}$, is then computed over the masks $r, r \in \mathcal{M}$:

$$\forall p \in r, r \in \mathcal{M} \qquad \boldsymbol{\omega_{res}}(p) = \boldsymbol{\omega_{inp}}(p) - \boldsymbol{\omega}(p) \tag{4.5}$$

The residual flow is illustrated in Figure 4.10.

From the residual flow, we want to get a motion saliency map g, whose values are scalar and defined within $[0, 1]$. To this end, we define g as follows :

$$\begin{aligned} \forall p \in \Omega \qquad & g(p) = 1 - \exp(-\lambda ||\boldsymbol{\omega}_{res}(p)||_2) \qquad \text{if } p \in r, r \in \mathcal{M} \\ & g(p) = 0 \qquad \text{otherwise} \end{aligned} \tag{4.6}$$

where λ weights the saliency score. Function g expresses that non-zero residual motion highlights salient moving elements. Parameter λ allows us to establish a trade-off between robustness to noise and ability to highlight small but still salient motions.

Let us note that, if we were interested in a hard segmentation of independently moving objects, we would just need to set λ to a high value. Indeed, by applying a threshold τ to $g(p)$, we can deduce from (4.6) that p will be segmented if :

$$||\boldsymbol{\omega}_{inp}(p) - \boldsymbol{\omega}(p)||_2 \geqslant -\frac{ln(1-\tau)}{\lambda}. \tag{4.7}$$

With τ arbitrarily set to $\frac{1}{2}$ (the middle value of $[0, 1]$), the decision depends only on λ. Pixels with residual flow magnitude greater than $\frac{ln(2)}{\lambda}$ will be segmented as salient. This shows that our method is flexible, since we can shift from the motion saliency problem to the video segmentation problem just by tuning parameter λ.

4.2.4 Bidirectional processing

Finally, we propose to further leverage the temporal dimension to refine the saliency map. To do this, we introduce a bidirectional processing (see Figure 4.1 a)-g)). The whole workflow is applied twice in parallel, backward and forward, that is, to the image pair $I(t), I(t-1)$ and the image pair $I(t), I(t+1)$. This yields two motion saliency maps, respectively g_b and g_f, such as the ones that are represented in Figure 4.11. We combine these maps by taking their pixel-wise minimum to reduce the number of false positive. This gives us the final saliency map $g_{\mathcal{F}}$, as illustrated in Figure 4.12 :

$$g_{\mathcal{F}}(p) = \min(g_b(p), g_f(p)). \tag{4.8}$$

The figure shows that areas close to motion boundaries are refined, as for instance for the *soapbox* case. Accordingly, the road pixels between the man and the handle are not highlighted as salient. We can also observe that areas that have an unstable apparent motion can be discarded this way. This is the case for instance for the *Libby* video. The background is close to the moving camera, and the bidirectional processing made it possible to clean up a significant part of the final saliency map.

The reported experimental results for MSI-ns, MSI-fm, MSI-pm, and for the NM method introduced in Section 4.3.3, will include the bidirectional processing.

FIGURE 4.4 – Sample videos processed in the experiments illustrating a variety of scenes. Only one frame in each video is displayed.

FIGURE 4.5 – a) Optical flows computed between two successive images of the videos illustrated in Figure 4.4. The backward element is on the left and the forward on the right for each sample.

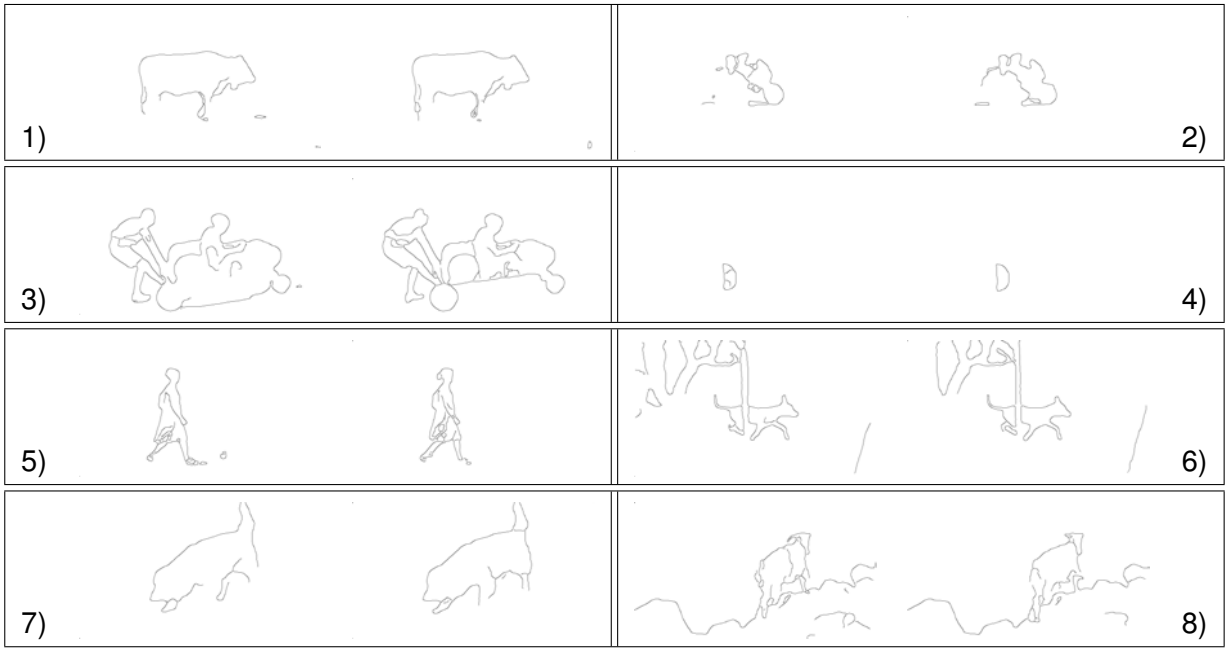


FIGURE 4.6 – b) Motion boundaries computed on the flow fields of Figure 4.5. The backward element is on the left and the forward on the right for each sample.

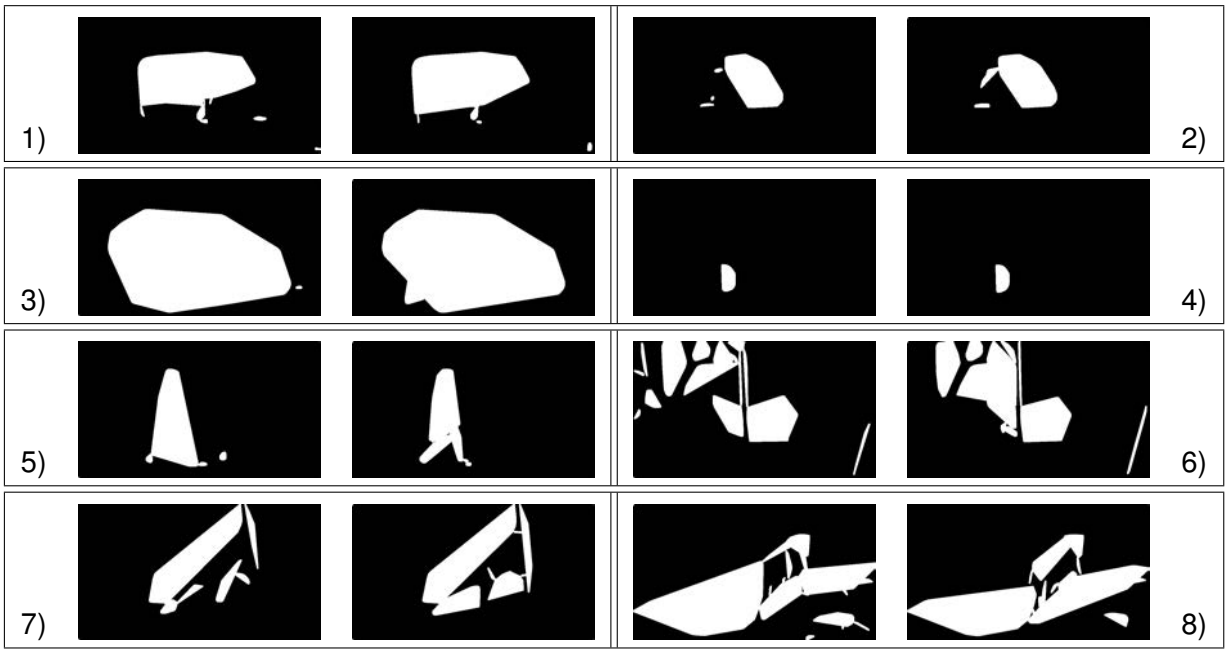


FIGURE 4.7 – c) Dilated convex envelopes obtained from the contours shown in Figure 4.6. The backward element is on the left and the forward on the right for each sample.

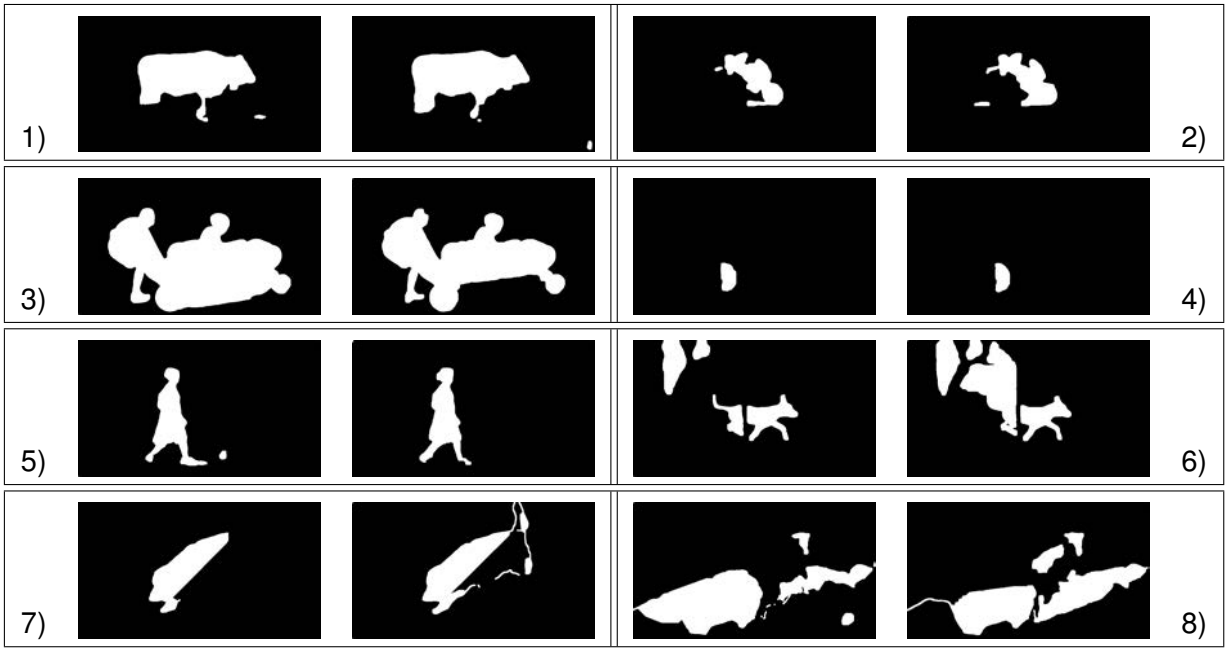


FIGURE 4.8 – d) Inpainting masks issued from the envelopes of Figure 4.7. The backward element is on the left and the forward on the right for each sample.

FIGURE 4.9 – e) Inpainted optical flows (with the border of the inpainting masks superimposed). The backward element is on the left and the forward on the right for each sample.

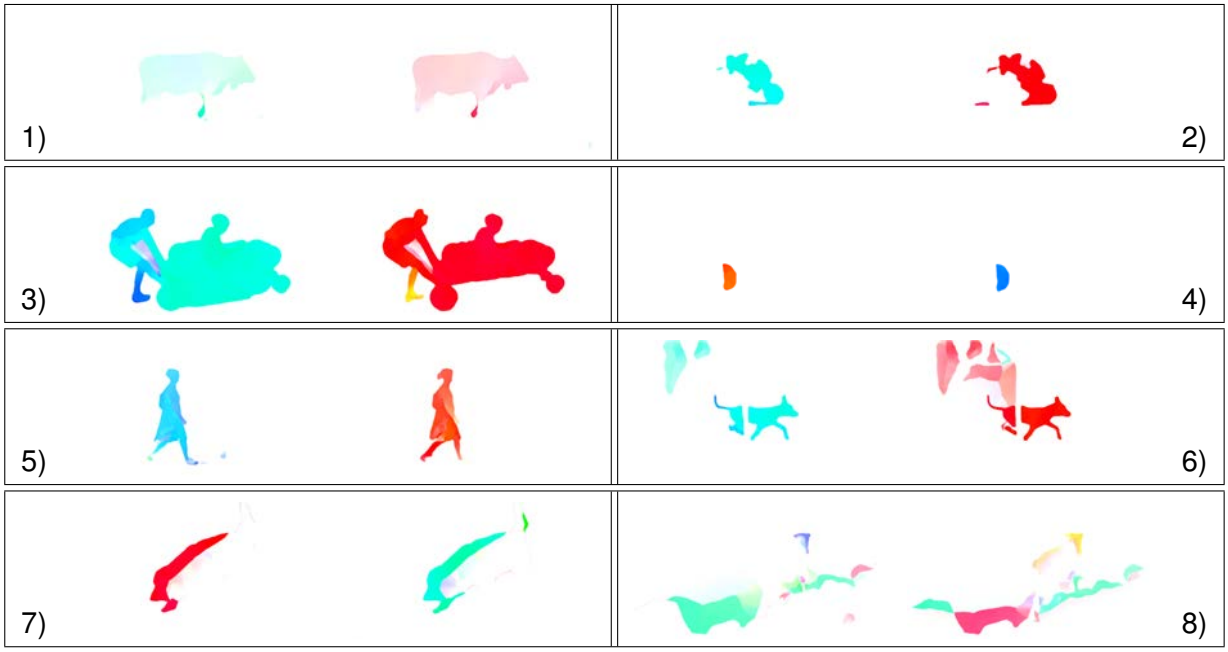


FIGURE 4.10 – f) Residual flows. The backward element is on the left and the forward on the right for each sample.

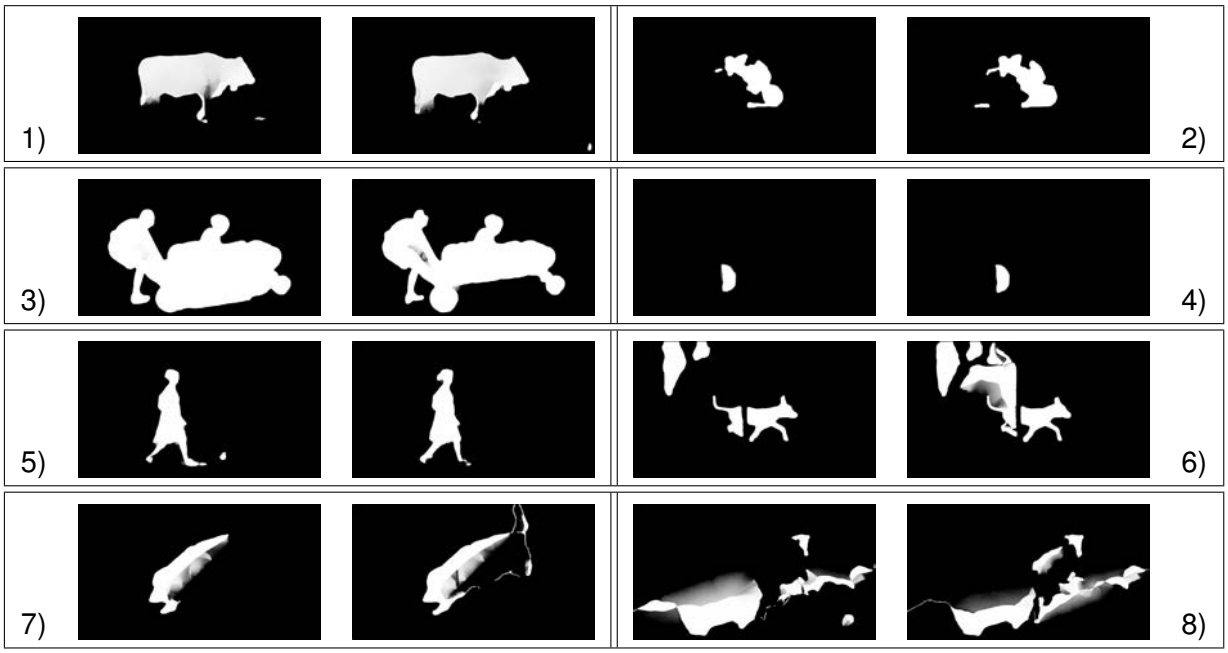


FIGURE 4.11 – g) Forward and backward motion saliency maps corresponding to the residual flows of Figure 4.10.

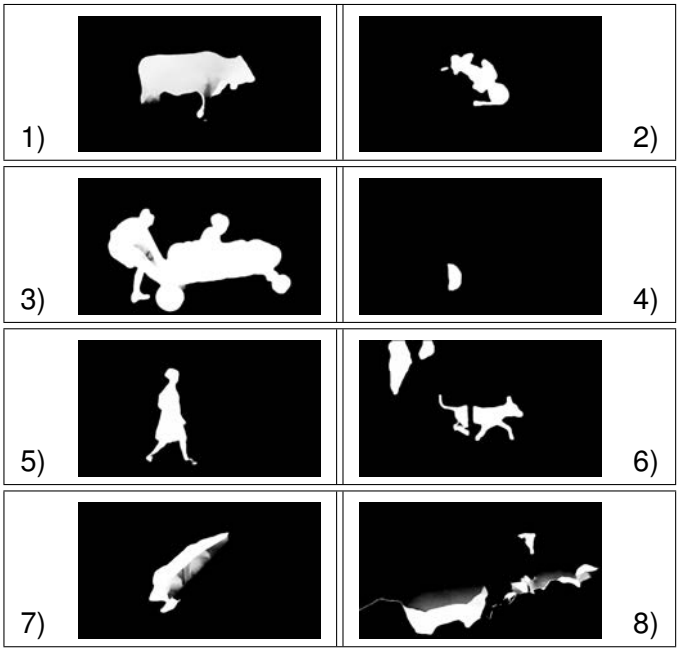


FIGURE 4.12 – h) Final motion saliency maps computed from the bidirectional maps of Figure 4.11.

4.3 Experimental results

We will first give additional information regarding the experimental setting in Section 4.3.1. We will then report quantitative results in Sections 4.3.2 and an ablation study in Section 4.3.3. Finally, Section 4.3.4 will present a qualitative evaluation.

4.3.1 Experimental setting

The optical flow is the foundation on which our method is built. The optical flow method should then satisfy several properties to make the results satisfying enough. First, the optical flow should be as accurate as possible, while being computed as quickly as possible to make the processing of whole video sequences tractable. In Section 2.4, we observed that deep learning-based methods have recently supplied state-of-the art results while running practically in real time. FlowNet 2.0 [Ilg+17] and PWC-Net [Sun+18] are examples of this class of methods for which pre-trained networks are publicly available.

Here, let us briefly describe the FlowNet 2.0. algorithm. Its architecture, described in Figure 4.13, consists in a stack of convolutional networks that progressively refines the optical flow. The authors train the networks one after the other to avoid over-fitting. The endpoint error (EPE) can then be used as a training loss for this network. It is defined as follows (with ω_{GT} the ground truth optical flow and ω the estimated flow) :

$$\text{EPE} = \frac{1}{|\Omega|} \sum_{p \in \Omega} ||\omega(p) - \omega_{GT}(p)||_2 \tag{4.9}$$

The PWC-Net network relies on similar principles, with notable differences being that it warps learned features instead of the images, and that it has significantly fewer parameters than FlowNet 2.0.

For our method, another important issue is in the preservation of sharp motion discontinuities. Indeed, we rely on motion boundaries to extract the inpainting masks. Figure 4.14 contains examples of flows computed with FlowNet 2.0 and PWC-Net, with trained models provided by the authors. We observe that the flows estimated with FlowNet 2.0 are sharper compared to the flows computed with PWC-Net. Despite the superior performance of PWC-Net on the MPI-Sintel benchmarks [But+12], we accordingly decided to adopt FlowNet 2.0 for the estimation of the optical flow fields. Since the optical flow estimation is a research topic that is still quickly evolving, replacing FlowNet 2.0 with a more accurate method producing sharp motion boundaries is expected to improve our process for motion saliency estimation.

For all the experiments, the parameters of our framework are set as follows. The Canny edge detector is applied to the image smoothed with a Gaussian filter of standard deviation $\sigma = 5$. The two thresholds for the Canny edge detector are set to 20 and 60. The ratio of 3

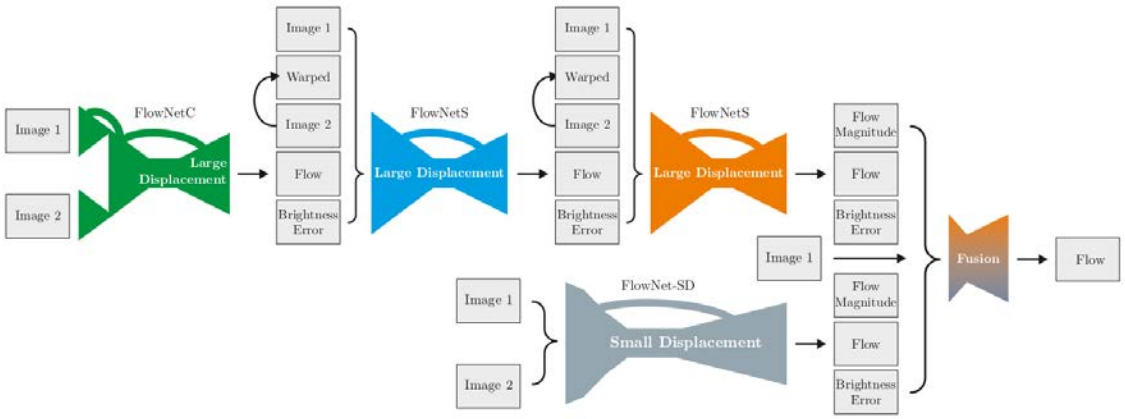


FIGURE 4.13 – Architecture of FlowNet 2.0. Reproduced from [Ilg+17].

between the two values is chosen following the advice of [Can86]. For the inpainting algorithm, a radius of 5 pixels around the region to inpaint is used. Let us note that this radius of 5 pixels as well as the choice of a 5x5 kernel for dilation operations have been used for images of various dimensions. The images of the DAVIS dataset have a size of 854x480 pixels, and the additional examples presented in Figure 4.15 have a size of 720x720 pixels for the generated example and 352x288 for the infrared video. In fact, the value of a radius of 5 pixels and of a 5x5 kernel are expected to be adapted as long as they allow to cover a local neighbourhood. Indeed, the role of the dilations is simply to be robust to local noise, and the inpainting relies on the local value at the boundary of the inpainting area. This setting should be adequate for common image resolutions, except maybe for very large images (height and width of thousands of pixels) for which a larger radius may be selected. On the other hand, a radius of 5 pixels may be too large for very small images (height and width of a few dozen of pixels), and in this case a value of 3 may become preferable.

Finally, the parameter λ for the computation of the saliency map has been set to $\frac{3}{2}$. It appeared adapted to highlight the salient moving objects, while putting less emphasis on weaker motions.

There is no available benchmark explicitly dedicated to motion saliency. Therefore, we chose the DAVIS 2016 dataset for the evaluation of our method. This dataset has been initially introduced in [Per+16] for the video object segmentation (VOS) task. It has also been recently used to evaluate methods estimating saliency maps in videos, as in [WSS18 ; LS18]. For the VOS task, the object to segment is a foreground salient object of the video, which has a distinctive motion compared to the rest of the scene. It makes this dataset exploitable for motion

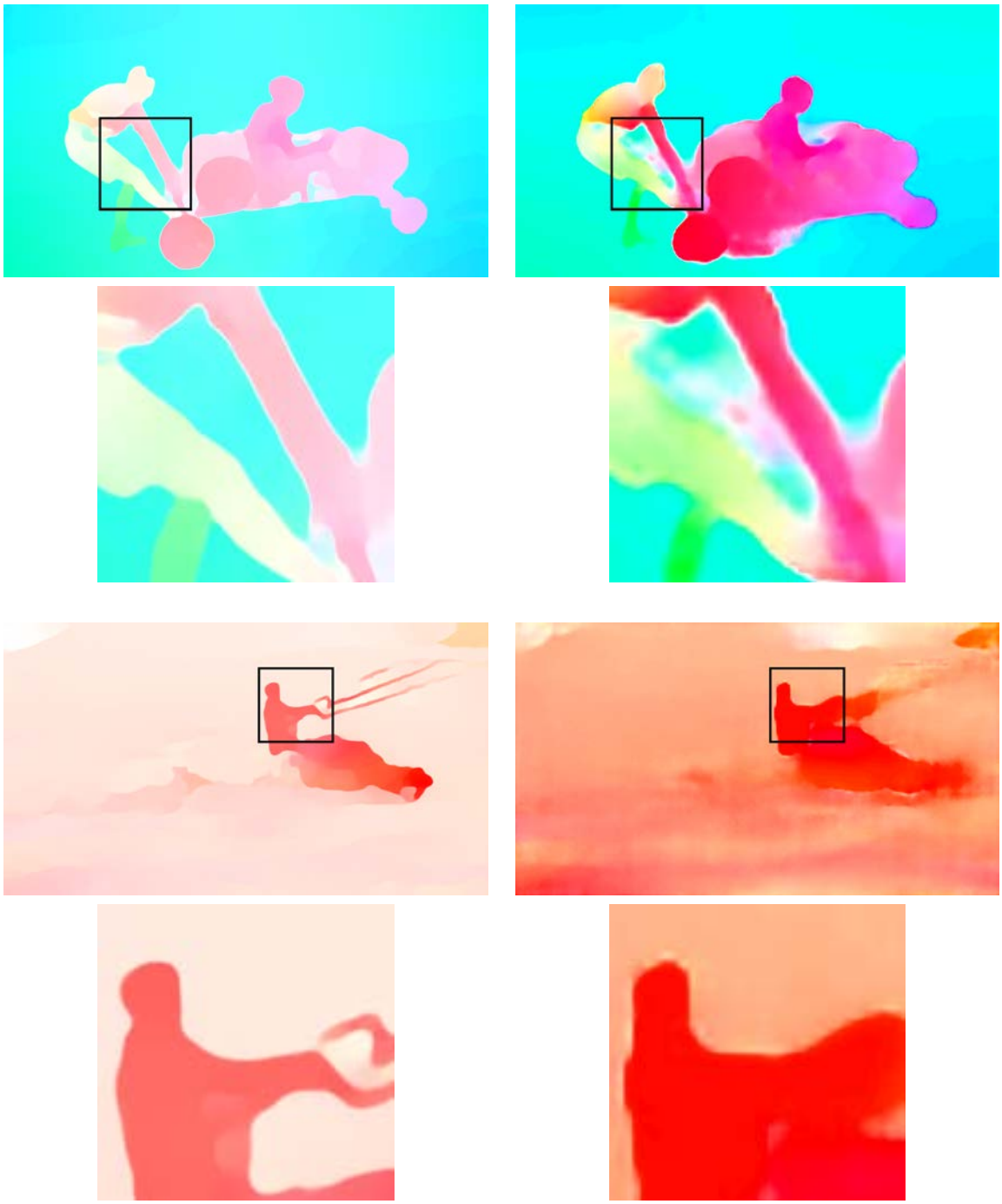


FIGURE 4.14 – Optical flow computed on the *soapbox* and *kite-surf* samples with FlowNet 2.0 (left) and PWC-Net (right). The zooms shows that the borders estimated with FlowNet 2.0 are sharper.

saliency estimation, although appearance plays a role.

4.3.2 Quantitative comparison

Table 4.1 collects comparative results of our three variants, MSI-ns, MSI-pm and MSI-fm, with state-of-the-art methods for saliency map estimation in videos : LGFOGR [WSS15], SAG [WSP15], RST [LS16], STCRF [LS18] and VSFCN [WSS18]. These methods have been presented in Section 2.5 of Chapter 2. We have collected the performances of these methods from [LS18], except for [WSS18], for which we used saliency maps provided by the authors to compute the metrics. We carried out the experimental evaluation on the test set of DAVIS 2016, that contains 20 videos. The quantitative evaluation on the DAVIS 2016 dataset is useful, but may generate a (small) bias since it is oriented to video segmentation. The available ground truth on DAVIS 2016 may not fully fit the requirements of the motion saliency task as illustrated in Figure 4.15 and commented in Section 4.3.4, since it is object-oriented and binary.

For the evaluation, we use the Mean Absolute Error (MAE), F-Adap and F-Max metrics that we compute the same way as in [LS18]. The MAE is a pixel-wise evaluation of the valued saliency map g (computed with equation 4.6) compared to the binary ground truth map s. The MAE is defined as follows :

$$\text{MAE} = \frac{1}{N} \sum_{p \in \Omega} |g(p) - s(p)| \tag{4.10}$$

with p a pixel of the image domain Ω, and N the number of pixels in the image.

F-Adap and F-Max are based on the weighted F-Measure. This metric requires a binary saliency map, and the difference between F-Adap and F-Max lies in the way the binarisation is obtained as explained below. The weighted F-Measure is defined as follows, with the weight β^2 being set to 0.3 following [LS18] :

$$F_\beta = \frac{(1+\beta^2) Precision \times Recall}{\beta^2 \times Precision + Recall}. \tag{4.11}$$

Note that F_β is defined this way to provide an evaluation measure considering recall β times as much important as precision [VR79].

F-Adap involves an adaptive threshold $\tau = \mu + \sigma$ to binarise the saliency maps. The values μ and σ are computed for each saliency map, and they correspond respectively to the mean and standard deviation of the saliency values. F-Max is the maximum of the F-Measure for thresholds varying in [0,1], or equivalently in [0,255] if the saliency map is converted to a grayscale image of integers.

Our method MSI-ns obtains consistently satisfactory results, as it ranks second for the three metrics. The two other variants, MSI-pm and MSI-fm, respectively rank third and fourth, but follow MSI-ns by a small margin. Let us recall that we obtain our results without any learning

on saliency and any appearance cues in contrast to [LS18], which performs the best. Our parametric and diffusion-based flow inpainting methods have close performance on the DAVIS 2016 dataset. However, the latter should be more easily generalisable, since the background motion cannot be always approximated by a single parametric motion model. If we restrict the parametric model to the surrounding area, then, the issue of specifying the latter will arise.

Regarding the computation time, the MSI-ns method takes 1.2 seconds to estimate the motion saliency map for a 854x480 frame on a 2.6 GHz processor. Compared to the computation time we reported in [MBL19b], the code has been re-shaped, which explains the improvement. Our code is written in Python and can be further optimised. Notably, the forward and backward streams of the workflow can be parallelised.

Method	MAE ↓	F-Adap ↑	F-Max ↑	Appearance	Motion	Supervised
STCRF [LS18]	**0.033**	**0.803**	**0.816**	*Yes*	*Yes*	*Yes*
MSI-ns	<u>0.043</u>	<u>0.735</u>	<u>0.751</u>	*No*	*Yes*	*No*
MSI-pm	0.044	0.724	0.750	*No*	*Yes*	*No*
MSI-fm	0.045	0.716	0.747	*No*	*Yes*	*No*
VSFCN [WSS18]	0.055	0.698	0.745	*Yes*	*Yes*	*Yes*
RST [LS16]	0.077	0.627	0.645	*Yes*	*Yes*	*No*
LGFOGR [WSS15]	0.102	0.537	0.601	*Yes*	*Yes*	*No*
SAG [WSP15]	0.103	0.494	0.548	*Yes*	*Yes*	*No*

TABLE 4.1 – Comparison with state-of-the-art methods for saliency map estimation on the test set of DAVIS 2016. In bold, the best performance, underlined, the second best. We also indicate whether the method relies on appearance information, on motion information, and whether it is supervised.

4.3.3 Ablation study

In this section, we present an ablation study of our method, and we investigate two main points.

First, we introduce a naive method (named NM) to motion saliency estimation to better assess the contribution of the main components of our method. It merely consists in first computing the dominant (or global) motion in the image. To this end, we estimate an affine motion model with the robust multi-resolution algorithm Motion2D [OB95]. No inpainting masks are extracted. The residual flow contributing to the motion saliency map is now the difference, over the whole image, between the computed optical flow and the estimated parametric dominant flow. In comparison, the variant MSI-pm computes the residual flow similarly, but the estimation is restricted to the inpainting masks. This setting is intended to evaluate the role of the whole process that consists in finding and leveraging inpainting masks.

Results are reported in Table 4.2. We can see that the NM method yields poor performances. This demonstrates that extracting candidate masks and applying optical flow inpainting to these regions only is necessary for the successful application of our method.

The second point we investigate is the role of the bidirectional processing. To do so, we apply MSI-ns, MSI-fm and MSI-pm without the bidirectional processing, by considering the forward stream only. The final saliency map is then the forward saliency map. By comparing the results with the standard variants of our method in Table 4.2, we see that the bidirectional processing consistently improves the performances.

Method	MAE ↓	F-Adap ↑	F-Max ↑
MSI-ns bidirectional	**0.043**	**0.735**	**0.751**
MSI-pm bidirectional	<u>0.044</u>	<u>0.724</u>	<u>0.750</u>
MSI-fm bidirectional	0.045	0.716	0.747
MSI-ns forward	0.047	0.713	0.735
MSI-pm forward	0.051	0.690	0.730
MSI-fm forward	0.048	0.701	0.735
NM	0.453	0.367	0.612

TABLE 4.2 – Ablation study of our method, with results on the DAVIS 2016 dataset. In the first three rows, we recall the performance of the standard variants of our method. Then, we present the results without the bidirectional processing (only the forward stream is considered). Finally, the method NM is included. In bold, the best performance, underlined, the second best.

4.3.4 Qualitative evaluation

We complement the quantitative evaluation presented in Section 4.3.2 by a visual qualitative evaluation. For this, we consider our method MSI-ns, which turns out to be the best of the three variants as reported in Table 4.1. Figure 4.15 displays the output of our method for frames of the videos *soapbox, Lucia, cows, bear, motorbike, soccerball, kite-surf, Libby, dog* and *goat* of the DAVIS 2016 dataset, and for two other types of videos. In the eleventh example (*lawn* video), a rectangular region in the lawn was artificially moved in the image as indicated by the ground truth. It provides us with an example where the only discriminative information is supplied by the undergone motion. The twelfth image comes from the *park* video of the changedetection.net dataset [Goy+12]. It was acquired with a thermal camera, providing us with an example where appearance is of limited help.

Computed motion saliency maps and residual flows are shown in Figure 4.15 c)-d). The residual flow, although an intermediate step in our method, is meaningful on its own. It provides valuable additional information about the direction and magnitude of salient motions in the video. It can be considered as an auxiliary map to the saliency map.

For the *soapbox* and *Lucia* video clips, the salient elements with clearly distinctive motions have been almost perfectly extracted. The *cows* example exhibits an interesting behaviour. The cow is globally moving, except for its legs which are intermittently static. This illustrates the difference between the video object segmentation task, for which the whole cow should be segmented, and the motion saliency estimation task, for which the elements of interest are elements with distinctive motion. Consistently, our method does not involve the two legs in the saliency map. The *bear* displays a similar behaviour, with the salient element globally found with no highlighting on the almost static parts.

The *motorbike* and *soccerball* clips include a salient moving object partly occluded. In both cases, the method is able to properly find the visible part of the salient moving object.

In the *kite-surf* clip, the sea foam has a non rigid but strong motion, and consequently, it is likely to belong to the salient moving region. Consistently, our method assigns a high saliency value to the sea foam. In contrast to the VOS task, the kite-surfer is the only foreground object to segment as defined in the ground truth.

The *Libby* clip is a challenging case. On one hand, our method is able to properly find the dog despite the occluding pole. On the other hand, this video has been taken with a moving camera, relatively close to static objects in the background (the trees, the pole, the grid). This generates a large apparent motion for these elements, which are partly classified as salient.

The *dog* and *goat* represent failure cases. For the *dog* clip, the small magnitude of the dog motion made the motion boundaries harder to find and to link. For the *goat* clip, the rocks have a strong apparent motion due to the camera displacement, which made the motion of the goat comparatively less apparent. We will come back to the problems illustrated by the *Libby*, *dog* and *goat* clips in the conclusion.

For the last two video clips, saliency is only or mainly due to motion. In the *lawn* clip, the square region is easy to detect when seeing the video, but is impossible to localize in a single frozen image. Our method based on optical flow is able to recover the salient moving region. Finally, in the *park* clip involving an infra-red video with less pronounced appearance, our method also yields a correct motion saliency map.

4.4 Conclusion

Before concluding, let us discuss two limitations of the method we have defined. The first limitation concerns scene depth discontinuities. First, let us recall that the 2D apparent motion in the image results from both the relative 3D motion between the scene and the camera and the depth of the viewed objects. Then, motion discontinuities in the image may be due also to depth discontinuities in the scene. Accordingly, salient moving regions extracted from optical flow as we did, may correspond to static objects in the foreground when the camera is moving.

Indeed, these foreground objects exhibit a larger apparent motion than the rest of the static background. We will call this specific configuration “depth saliency”. However, our framework based on optical flow, that is, the apparent motion in the image, makes no distinction between “true” motion saliency and depth saliency, as illustrated by Figure 4.16. In a scene with depth saliency only, we can overcome this problem by first applying our supervised method for motion saliency detection presented in Chapter 3. We can then estimate saliency maps only if the frame is labelled as dynamically salient. Indeed, we demonstrated that our image-based saliency detector is robust to depth saliency, certainly owing to the supervised learning stage. Yet, distinguishing depth and motion saliency when both are present in the very same image remains an open problem.

The second limitation is related to the scenario where there are three groups in the scene : a moving salient element, a larger number of moving elements that undergo “normal” motion, and the static background. This can correspond for instance to a flow of pedestrians going in one direction, while one pedestrian is moving against the flow (see Figure 4.16). Depending on the locations occupied by the three groups of motion in the image, our method will not be able to correctly identify salient motions. For instance, normal moving objects might be declared as salient, if they are circumvented by the static background in the image. Properly handling this situations of relative saliency, that is, salient moving object against normal moving objects, will be among the objectives of the next chapter.

As a conclusion, we have proposed in this chapter a new paradigm to estimate motion saliency maps in video sequences based on optical flow inpainting. It yields valued saliency maps to highlight and locate the presence of motion saliency in videos. We tested our method on the DAVIS 2016 dataset. We obtained state-of-the-art results, while using only motion information and introducing no learning stage. This makes our method of general applicability. Additionally, the computed residual flow on its own provides additional information on motion saliency, which could be further exploited.

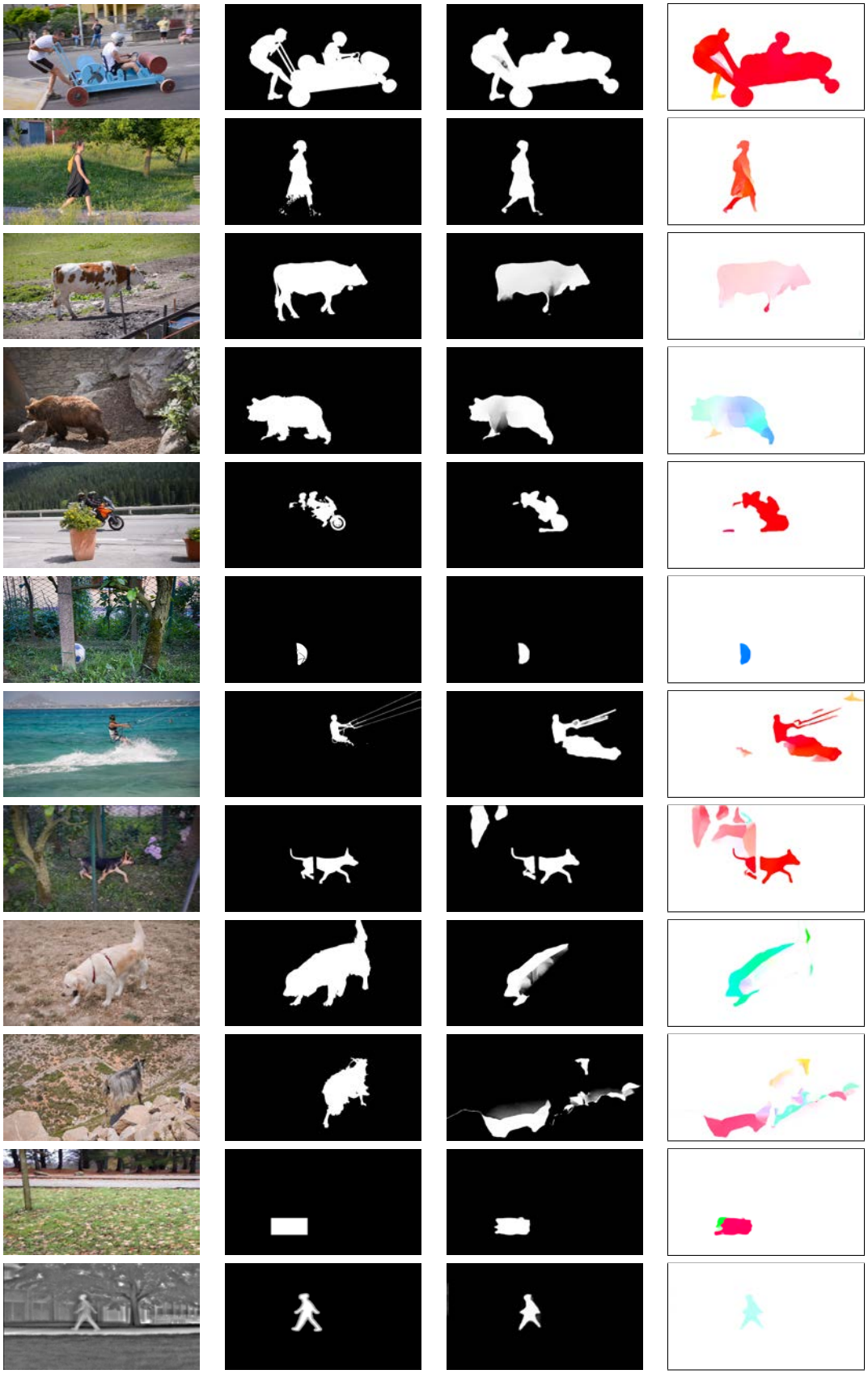


FIGURE 4.15 – From left to right : a) one image from the video, b) binary ground truth, c) motion saliency maps predicted by our method MSI-ns, and d) the estimated forward residual flow (displayed with the motion colour code of Figure 4.2).

FIGURE 4.16 – Valued motion saliency map in presence of depth discontinuity in the scene. The traffic sign stands in the foreground, and consequently, has a different apparent motion (larger motion) due to the moving camera than the rest of the static background. Accordingly, it is found salient by MSI-ns, while it is a static object. Note that this frame is classified as not salient by the motion saliency detector presented in Chapter 3.

FIGURE 4.17 – Valued motion saliency map in presence of three groups of apparent motion : the static ground, a flow of pedestrians moving from bottom to top and one salient pedestrian moving from top to bottom. The presence of the three groups of apparent motion confuses our method, which labelled as salient a portion of the ground (left element), the salient pedestrian (middle element) and a non-salient pedestrian together with another portion of the ground (right element).

CHAPTER 5

DETECTION OF SALIENT TRAJECTORIES IN VIDEOS FROM A LEARNED LATENT REPRESENTATION

5.1 Introduction

This chapter is concerned with the estimation of trajectory saliency. Trajectory saliency naturally allows one to consider progressive motion saliency over time. More specifically, we are talking about motion saliency that is not perceptible instantaneously but that progressively appears over time. It can again be useful to trigger alarms, to allocate additional processing or to allow for the detection of a specific event. Trajectories are the most natural features to support progressive dynamic saliency detection. Accordingly, we will talk about *trajectory saliency*. Indeed, trajectory-based anomaly detection emerges as a growing need in many applications, possibly under different names [RB19 ; Mou+15 ; Fan+18]. This consists in identifying, among a group of moving elements, the few that may follow a distinct motion pattern compared to the collective motion.

To solve this problem, we propose an approach based on the estimation of a latent representation of trajectories. Two main issues arise :

— Which representation for the trajectories ?

— Which method for trajectory saliency detection ?

We aim to design a method as general and efficient as possible. To do this, learning-based approaches seem nowadays superior to hand-crafted representations. As we are not limited by the amount of data, we can learn the trajectory representation with neural networks. Besides, we will design a decision algorithm adapted to the desired properties of the learned representation to solve the second issue. This representation is learned with a constrained auto-encoder network. For this, we introduce a consistency constraint in the training stage, which expresses that, in a given scenario, most trajectories are expectedly normal and should then have a similar representation.

The rest of this chapter is organised as follows. Section 5.2 presents our latent trajectory

representation based on a LSTM auto-encoder. Section 5.3 deals with the detection of trajectory saliency. In Section 5.4, we present the datasets we use. In Section 5.5, we give all the implementation details, and in Section 5.6 we report the experimental results including an objective comparative evaluation. Section 5.7 provides further results to validate design choices of our method. Section 5.8 proposes complementary visualisation and experiments. Finally, Section 5.9 contains concluding comments.

5.2 Latent trajectory representation

Our objective is to detect salient trajectories within a set of trajectories, the vast majority of which are normal trajectories. In the following, a scenario $\mathcal{S}$ will consist of a set of normal trajectories for a given context and possibly a few salient trajectories with respect to the same context. Indeed, a trajectory is not salient in itself, but with respect to a given context. Salient trajectories are supposed to depart from the motion pattern shared by the normal trajectories.

First, we need a relevant representation of the trajectories. It must be compact to form the basis of an efficient classification method, while descriptive enough to distinguish normal and salient trajectories. It is preferable to transform trajectories of any length into a constant-length representation, in order to enable adaptability to any scenario. It should be applicable to both 2D and 3D trajectories. In that vein, hand-crafted representations based on sophisticated mathematical developments were investigated in the past, as in [HBL08]. The latter aimed to get representations invariant to a large class of geometrical transformations, including translation, rotation, scale, and able to achieve fine classification. However, our goal is different here. In addition, it is now more tractable to learn the targeted transformation with the advent of neural networks, providing enough data are available for training [Su+16 ; Yao+17]. An appropriate means is then the use of auto-encoders. The internal (latent) code in the auto-encoder will provide us with the desired representation. Besides, a trajectory exhibits a specific nature inherent to the time dimension, making a recurrent network a suitable choice. Below, we will first describe the representation we have designed for trajectories based on these requirements. Then, in Section 5.3 we will explain how we use it to detect salient trajectories.

5.2.1 LSTM-based network for trajectory representation

Our approach is to compute the trajectory representation with a recurrent auto-encoder network. More specifically, we adopt a Long Short-Term Memory (LSTM) network. The objective of an auto-encoder is to reconstruct its input, by relying on an intermediate compact representation [RLL18]. This intermediate representation is used as the code $c \in \mathcal{R}^n$ representing the input. An auto-encoder has the advantage of being unsupervised, since no manual annotation

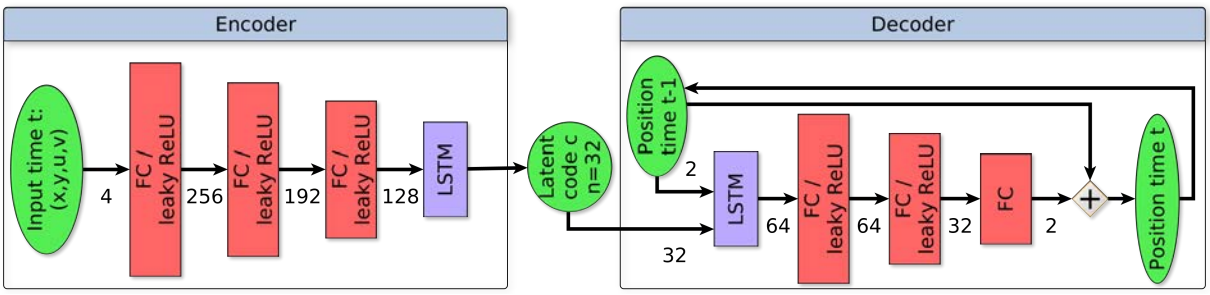


FIGURE 5.1 – Recurrent auto-encoder network computing the latent representation of trajectories. Numbers indicate the input dimension for each layer.

is required for training.

This architecture is inspired from existing networks, such as the generator of [AHP19]. We similarly build a stack of fully connected layers to process the data at each time instant, but we dedicate the temporal processing to the recurrent LSTM units. This approach allows us to handle trajectories of variable length. In contrast, a network involving 1D temporal convolutions instead of temporal recurrence would require additional processing to feed the network with a constant-length and adequate representation of the whole trajectory.

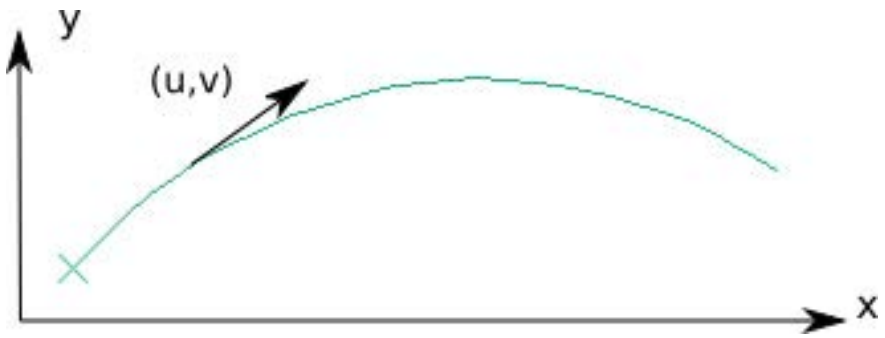


FIGURE 5.2 – A trajectory in the 2D-space. (x,y) denotes the position along the trajectory, and (u,v) the velocity.

The architecture of the auto-encoder supplying the trajectory representation is summarised in Figure 5.1. The input of the network at time t is composed of the concatenation of the 2D position and velocity vectors constituting the trajectory at time t. For the processing of the whole trajectory $\mathcal{T}_i$, it gives a total of 4τ features if τ is the length of the trajectory, i.e., the number of points it contains. The four-component individual vector is given by (x_t, y_t, u_t, v_t), where (x_t, y_t) is the point position in the 2D-space and (u_t, v_t) the velocity vector at time t as illustrated in Figure 5.2. In practice, we assimilate the velocity to the displacement. Although the displacement can be deduced from two successive positions, it turns out to be relevant and it can be obtained at negligible cost. This is why we include it in the input vector. The 2D-space can be the image

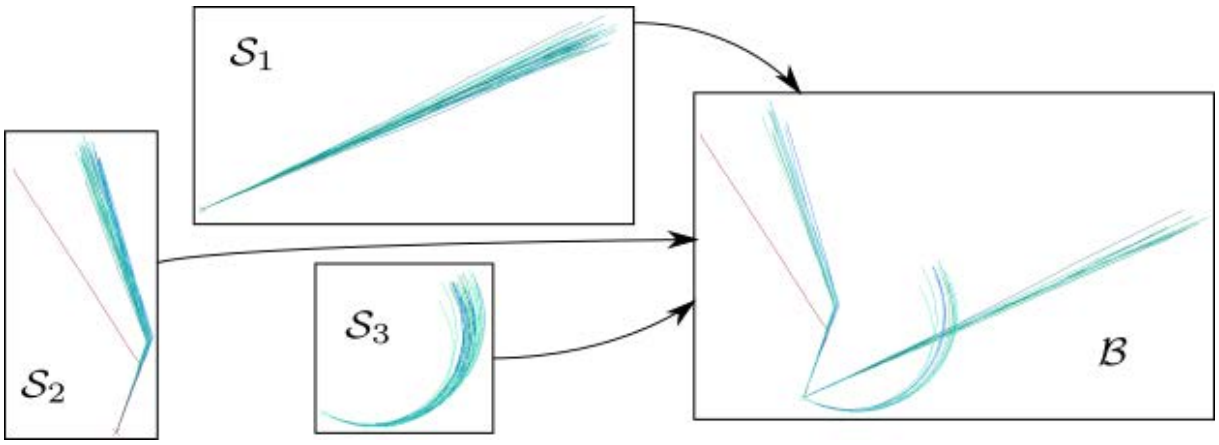


FIGURE 5.3 – Trajectories of three scenarios $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ are displayed on the left. On the right, a batch $\mathcal{B}$ is represented that is built with trajectories sampled from the three scenarios.

plane and the velocity vector given by the optical flow. If the 3D movements are planar, the 2D-space could be the ground plane of the 3D scene and trajectories (positions and velocities) are referred to this plane. An extension to full 3D trajectories would be straightforward.

The exact number of layers and their dimensions have been set during preliminary experiments. In particular, we set the code dimension to $n = 32$, as this value allows us to store enough information to represent the motion patterns, while being shorter that the typical trajectory representation of length 4τ. In the experiments, we consider trajectories with a number of positions that varies between 20 to 200 approximately.

To process one time step, the encoder activates a succession of three fully connected layers with output of dimension 256, 192 and 128 respectively, with the leaky ReLU activation function. Then, a LSTM layer with an output of dimension $n = 32$ is applied to get the representation of the whole trajectory.

The decoder reconstructs the trajectory from code $c \in \mathcal{R}^n$, which is the code obtained when the trajectory has been fully processed by the encoder. A second LSTM network takes as input the code c concatenated with the position at time $t-1$, and produces a vector of 64 components. The decoder is only expected to expand the information compressed in the code. The predicted position for the current time instant is obtained through two fully connected layers, followed by the leaky ReLU non linearity, whose respective output dimensions are 64 and 32, and one final fully-connected layer of output dimension 2 representing the displacement. The position is given by the sum of the displacement and of the previous position. Indeed, preliminary experiments showed that predicting the displacement is more robust than predicting the position directly. Let us mention that, for the reconstruction of a trajectory, we assume that its length is known. In practice, it is a metadata associated to each trajectory. The number of parameters for the whole network is 127970.

To train the auto-encoder, we adopt the reconstruction loss $\mathcal{L}_r$, defined as follows :

$$\mathcal{L}_r = \sum_{\mathcal{T}_i \in \mathcal{B}} \sum_{t=t_{init}}^{t_{final}} (x_t^i - \hat{x}_t^i)^2 + (y_t^i - \hat{y}_t^i)^2, \tag{5.1}$$

with t_{init} and t_{final} the initial and final time instants of the trajectory $\mathcal{T}_i$, (x_t^i, y_t^i) its position at time t, $(\hat{x}_t^i, \hat{y}_t^i)$ the predicted position and $\mathcal{B}$ a batch of trajectories. A batch can be composed of trajectories taken from several scenarios as illustrated in Figure 5.3. It is a standard practice to train auto-encoders with the reconstruction error as loss function [Yao+17 ; RLL18]. Since the positions entirely define the trajectory, the loss $\mathcal{L}_r$ involves only positions.

Once trained, the auto-encoder is ready to provide a code representing any trajectory at test time. However, there is so far no guarantee that two similar trajectories will be represented with close codes. This is an important issue since we will exploit the codes to predict saliency. It motivates us to introduce a second loss term for code estimation.

5.2.2 Additional consistency constraint

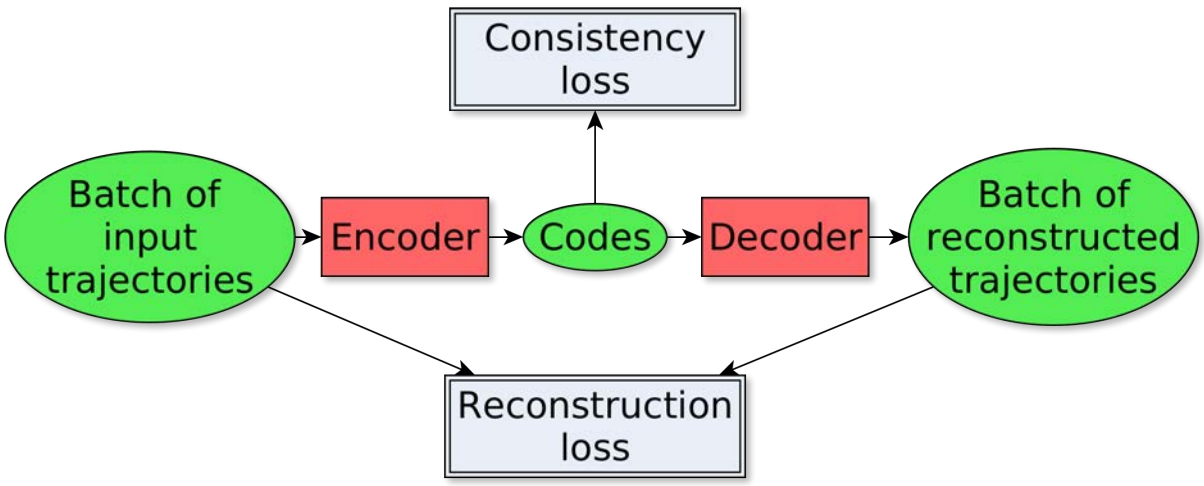


FIGURE 5.4 – Training scheme of the recurrent auto-encoder with the two associated losses.

Our objective is to represent similar trajectories with close codes and dissimilar trajectories with distant codes to make saliency prediction easier, taking inspiration from the paradigm of deep metric learning. We want to ensure the best possible consistency in the representation of normal trajectories, while drastically minimising, or even discarding, manual annotation. To this end, we complement our auto-encoder network with a consistency constraint. The consistency constraint takes the form of a second loss term, expressing our knowledge of the problem to solve. Figure 5.4 summarises our overall training scheme.

Let us recall that we address the problem of detecting salient trajectories that are supposed to be rare within a group of normal trajectories. The normal trajectories follow a consistent motion pattern. We then define the following consistency loss term $\mathcal{L}_c$ applied independently to each scenario $\mathcal{S}_k$ present in the batch $\mathcal{B}$:

$$\mathcal{L}_c = \sum_{\mathcal{S}_k \triangleright \mathcal{B}} \sum_{\mathcal{T}_i \in \mathcal{S}_k \cap \mathcal{B}} ||c_i - \widetilde{c}_k||_2, \tag{5.2}$$

with c_i the code of the trajectory $\mathcal{T}_i$ estimated by the network, and $\widetilde{c}_k$ the median code for the trajectories of the batch issued from the scenario $\mathcal{S}_k$. By using the $\triangleright$ operator, we mean the scenarios which contribute to the batch.

Applying the consistency loss to all the trajectories of the scenario means obviously that salient trajectories might be involved as well if there are any. On one hand, this setting may seem contradictory to our objective. However, let us point out again that salient trajectories are rare by definition. Therefore, they are unlikely to affect the value of the median code. In addition, the use of the L2 norm instead of the square of it in the consistency loss function (5.2) further limits the impact of the consistency constraint on the salient trajectories.

The full loss will add the quadratic loss term $\mathcal{L}_r$ defined in equation (5.1) to the consistency constraint of eq. (5.2). One of the roles of $\mathcal{L}_r$ is to mitigate the effect of $\mathcal{L}_c$ on salient trajectories, by preserving a correct enough reconstruction for salient trajectories, and consequently distinct enough codes. Then, the large imbalance between salient and normal elements is not an issue, unlike for supervised classification. On the contrary, it becomes an advantage for the correct training of our network by limiting the impact of salient trajectories on the consistency assumption. Finally, let us recall that, as mentioned in Section 2.2 of Chapter 2, previous work concluded that deep networks can reach good performance, even when a high proportion of the training data is mislabelled [Rol+17]. This shows the robustness of deep networks to a noisy learning signal, and it supports the idea of defining a loss with two terms that may be occasionally conflicting. More importantly, this setting has the great advantage of being unsupervised.

The final expression of the loss taking into account reconstruction and consistency terms is given by :

$$\mathcal{L} = \mathcal{L}_r + \beta \mathcal{L}_c, \tag{5.3}$$

where β is a weighting parameter to balance the impact of the two loss terms. In practice, samples of several scenarios could be included in the same batch, to increase the diversity of the trajectories at each training iteration as illustrated in Figure 5.3. This point will be specified for each experiment that we will carry out.

We have defined a method to estimate a latent code to represent trajectories. It is specifically designed to obtain close codes for similar trajectories and distant codes for potentially salient trajectories.

5.3 Trajectory saliency detection

Once trained, the recurrent auto-encoder provides us with codes of input trajectories. Most trajectories are supposed to be normal, i.e., appertaining to the same type of movement. Now, we have to define an algorithm to detect salient trajectories against normal ones. More specifically, we consider a scenario $\mathcal{S}$ of the test dataset, containing normal and possibly salient trajectories, $\mathcal{S} = \{\mathcal{T}_i\}$. The trajectories are represented by their respective latent codes $\{c_i, c_i \in \mathcal{R}^n\}$. The scenario $\mathcal{S}$ will be further defined when reporting each experiment. For clarity, this section will describe exclusively the final version of our algorithm. A more detailed discussion of the advantages of this version compared to several alternatives will be presented in Section 5.7.

5.3.1 Distance between codes

First, for comparison purpose, we need to characterise the normal class by a generic code. It will be used to classify trajectories into normal and salient ones. Due to the possible presence of salient trajectories in $\mathcal{S}$, the generic code will be given by the median of codes over $\mathcal{S}$, denoted by $\widetilde{c}$. Since we deal with a *n*-component code, we compute the component-wise median code. Each component of $\widetilde{c}$ is the median value of the corresponding components of the codes c_i over $\mathcal{S}$.

The classification of a trajectory $\mathcal{T}_i$, as normal or salient, will leverage the distance d_i between its code c_i and the median code $\widetilde{c}$:

$$d_i = ||c_i - \widetilde{c}||_2. \tag{5.4}$$

The distances related to normal trajectories are expected to be smaller than the ones related to salient trajectories. To make the comparison invariant to the distance range, the empirical average of the distances d_i, noted $\bar{d}$, and the standard deviation of the d_i, noted σ, are computed over scenario $\mathcal{S}$. The normalised distance q_i is then computed for each trajectory :

$$q_i = \frac{|d_i - \bar{d}|}{\sigma}. \tag{5.5}$$

With this definition, $q_i \in \mathbb{R}_+$, with values ideally close to zero for normal trajectories. The status of each trajectory will be inferred from the normalised distance q_i, as described below.

5.3.2 Saliency test

Coefficients q_i introduced in Section 5.3.1 account for the (possibly progressive) deviation from the normal motion of the scenarios for each trajectory $\mathcal{T}_i$. Accordingly, a natural way to

predict saliency is to assume that trajectories with a distance q_i greater than a given threshold λ are salient.

Before describing the procedure we selected to set λ, let us list some properties that the q_i coefficients are expected to follow. First, these coefficients are derived from codes estimated from a learned network. Depending on the final state of the network after learning, and especially if the training settings were modified (very different data domain or different hyperparameters for instance), the code distribution in the latent space is expected to change. Another point, that will be discussed in more detail in Section 5.7.1, is that the coefficients q_i will vary with the presence of salient trajectories.

To set an appropriate λ for a specific network, the safest way is then to probe a range of values on a dedicated validation set. This approach is the one we adopt in practice. We consider candidates values of λ in [0,5] sampled with a step of 0.05, and we select the best one for the validation set. The value of $\lambda = 5$ corresponds to 5 standard deviation with the definition of the q_i coefficients. It is then expected to correspond to very salient elements and to be a relevant upper bound. We refer to Section 5.7.2 for a discussion on an alternative, but that we did not select in the end.

5.4 Datasets

In this section, we present the datasets we used to train and evaluate our method. We have built a synthetic dataset of trajectories called STMS (Synthetic Trajectories for Motion Saliency estimation), and we have used the dataset made available by [ARF14], comprising pedestrian trajectories acquired in a railway station with a set of cameras over a long time period, that we will denote RS.

5.4.1 STMS dataset

Our synthetic trajectory dataset for motion saliency (STMS), contains three trajectory classes : straight lines, trajectories with sharp turns and circular trajectories. A noise of small magnitude is added to increase the variability of the trajectories.

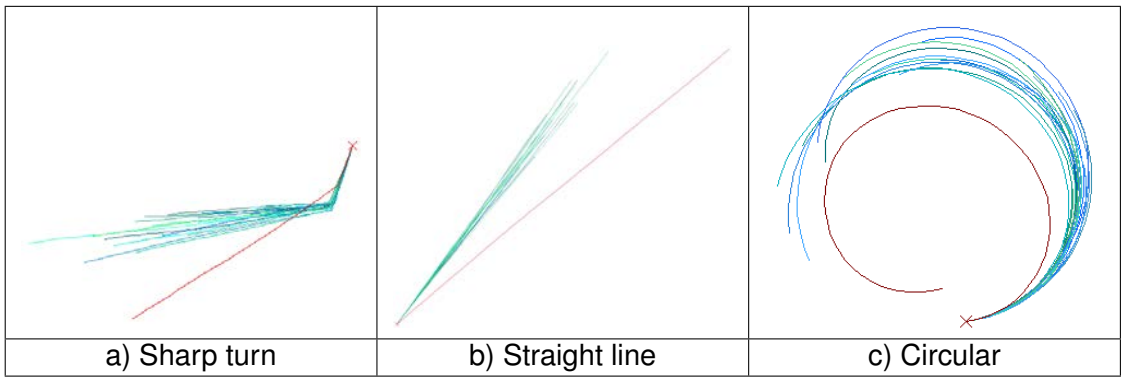
Each scenario is composed of trajectories of the same kind (e.g., straight line) and with similar parameters (velocity, initial motion direction, etc.). The number of positions is comprised between 20 and 60 for each trajectory. The trajectories have a velocity that varies numerically in $[5, 20]$ for different scenarios. The initial direction can vary in $[0, 2\pi]$ radians. For circular trajectories, the angular velocity is a constant comprised in $[-0.10, 0.10]$ radians per time step. For trajectories with sharp turns, rotations angles are comprised in $[-\frac{\pi}{2}, -\frac{\pi}{6}] \cup [\frac{\pi}{6}, \frac{\pi}{2}]$ radians for the turn. The initial position of the trajectories is set to the origin (0,0). With the turning time

for trajectories with sharp turns, these parameters fully define the backbone trajectory $\mathcal{T}_b$. For a given scenario, normal trajectories should all be similar, so only small variations around a constant value is allowed for these parameters. Still, to avoid having too uniform trajectories, a random noise $\mathcal{T}_n$ is added to each backbone trajectory. The position $(x_n(t), y_n(t))$ of $\mathcal{T}_n$ at time t depends on $(x_n(t-1), y_n(t-1))$ to ensure that $\mathcal{T}_n$ remains smooth. The final expression of the trajectories is then :

$$\begin{cases} x = x_b + x_n \\ y = y_b + y_n \end{cases}, \tag{5.6}$$

with (x, y) the position for the final trajectory $\mathcal{T}$ and (x_b, y_b) the position for the backbone trajectory $\mathcal{T}_b$.

To come up with a difficult enough task, salient trajectories are of the same class as normal trajectories. The difference will lie in the parametrisation of $\mathcal{T}_b$ for salient trajectories. The parameters are chosen so that the salient trajectory should be not too different from normal trajectories, but still distinct enough to prevent any visual ambiguity about its salient nature. An illustration is given in Figure 5.5.



a) Sharp turn | b) Straight line | c) Circular

FIGURE 5.5 – Samples of synthetic trajectories for the three classes, starting from the same origin. Salient trajectories are drawn in red, normal trajectories in green to blue.

The trajectories for training are generated on the fly. We generate scenarios composed of 10 normal trajectories. An additional salient trajectory may be added with a probability of 0.5. Accordingly, we build batches made of 6 scenarios each, for a total of 60 to 66 trajectories.

The validation set and the test set are composed of 500 scenarios each. For these two sets, each scenario is composed of 20 normal trajectories. An additional salient trajectory is included with a probability of 0.5. We did that to end up with a rate of salient trajectories of about 2.5%.

5.4.2 RS dataset

We now describe the RS dataset of real trajectories. We start with a general description of the dataset, before presenting the training procedure we applied to it.

5.4.2.1 Description of the dataset

The second dataset is constituted of real pedestrian trajectories, computed with a tracking system installed in a train station [ARF14]. The tracking system takes videos supplied by a network of cameras. The videos were processed to extract trajectories, which are projected on the 2D ground plane of the train station. More specifically, the trajectories at our disposal were extracted from two underground corridors, with gates on their sides and their extremities (one corridor is displayed in Figure 5.7). The acquisition was made during 13 days in February 2013, and resulted in a total of 115245 trajectories.

In this section, we will present the variants that we retained for the training on the RS dataset. These choices come from comparative experiments, that we will further detail at the end of this chapter in Section 5.8.4.

Compared to the STMS dataset, the trajectories of the RS dataset are longer, and the mean displacement between two points is larger (the mean numerical value of this displacements is 12.4 for STMS trajectories and 111 for RS trajectories, these values being obtained for validation subsets). Very long trajectories are challenging for recurrent networks such as the one we employ (its architecture was presented in Figure 5.1). To overcome it, we subsampled RS trajectories by keeping one every five points (consequently multiplying the observed displacement magnitude by a factor five). We also divided the coordinates by 100 to get displacement magnitudes closer to the synthetic case. The full trajectories are recovered by re-identification. This means that a given trajectory may be interrupted for some time, before being continued at a different point. To avoid these irregularities, we pre-processed the trajectories by splitting them when unexpectedly large displacements are observed.

For a better stability of the training, we found helpful to make all the trajectories start at the same origin (x_0, y_0). It can be viewed as a translation invariance requirement with respect to the location of the trajectory in the 2D-space. Accordingly, before any processing, we translate all the trajectories, so that their first point corresponds to the same given origin.

5.4.2.2 Training procedure

We pre-trained our network with the synthetic trajectories. We then considered two training procedures.

For the first training procedure, we leverage the available data to get a relevant consistency constraint. Indeed, the RS dataset involves many different motion patterns. We then build sce-

narios composed of normal trajectories sharing the same entrance and exit. It may happen that between a given entrance and exit, people take different paths. They can for instance go straight from the entrance to the exit, or they can make a detour to a given location in the train station as displayed in Figure 5.6. Moreover, we have no guarantee about the repartition of the trajectories between the different paths. It could for instance be possible to observe two different patterns represented by an equal number of trajectories, in which case none of these patterns would be salient, and the consistency constraint would not be applicable. To be as close as possible to a setting with only one normal motion pattern, we proceed as follows. We compute the median length of trajectories associated to a given entrance/exit pair. Only trajectories with a length deviating from less than 10% of the median length are kept. We build scenarios of 8 trajectories by following this procedure. Then, we draw 8 trajectories of the remaining set to form an elementary scenario. We group 8 elementary scenarios from several entrance/exit pairs to get the training batches. It improves the diversity of data at each iteration.

We consider three levels of consistency, respectively of $\beta = 10^5$, $\beta = 10^3$ and $\beta = 0$. The network variants trained with this procedure and with these consistency levels will be denoted respectively $V\beta_5$, $V\beta_3$ and $V\beta_0$ in the sequel. For this first training procedure, 43079 trajectories from the corridor PIW are included in the training set. Trajectories from the second corridor PIE were not included. Also, the algorithm used to get consistent batches discarded additional trajectories, such as the ones starting in the middle of the corridor, or the ones following rare paths. We have designed a second training procedure that leverages all the 95458 trajectories of the training set. For this procedure, we did not use the consistency constraint. It allowed us to build batches by selecting at random trajectories, thus increasing the diversity at each iteration. The batch size is set to 64 trajectories. We will denote this variant V_δ. This procedures allows us to evaluate how a network trained with more data and more diverse batches behaves, compared to a network trained with the consistency constraint and fewer data.

For both training procedures, training data come from the 11 first days of acquisition. We use the trajectories of the last two days of acquisition for validation and test.

The RS dataset was built for crowd analysis. There is no pre-defined saliency ground truth. Consequently, we have defined our own saliency experiments from the available data. We have exploited the organisation of the RS dataset into corridors. Let us stress that our goal is simply to be able to evaluate our method on real trajectory data. In particular, our point is not to investigate the RS dataset. It is then on purpose that we evaluate our method with evaluation settings, that will be for some of them constructed automatically from the available data. For these particular settings, we obviously do not expect to get additional information on the trajectories through the application of our method. We will describe these settings in Section 5.5.1.

5.5 Experimental setting

5.5.1 Evaluation settings for the RS dataset

We have designed three evaluation settings for the RS dataset that we will denote RSE-A, RSE-B and RSE-C, as described below.

5.5.1.1 Evaluation setting RSE-A

For the evaluation setting RSE-A, normal trajectories are those starting from the same given entrance and ending at the same given exit. To ensure that normal trajectories follow a similar motion pattern, we resort to the same pre-processing step as the above mentioned one used for the training stage and described in Section 5.4.2. In practice, it is sufficient to clean up the data, as illustrated in Figure 5.6. Salient trajectories have either a different entrance or a different exit, or both.

By following this procedure, we build a test set comprising 2275 normal trajectories. They are divided into 45 scenarios, corresponding to 45 different entrance/exit pairs. In each scenario, salient trajectories are trajectories corresponding to a different entrance/exit pair. To draw the salient trajectories from the available data, we considered three cases, from the easiest to the most difficult one :

— In the easiest case, salient trajectories are randomly drawn from entrance/exit pairs different than the one of the normal trajectories. Furthermore, only entrances/exit pairs leading to distinct motion pattern are allowed. For instance, if normal trajectories go from gates 7 to 8, salient trajectories cannot go from gates 9 to 10 since the trajectories would be parallel and of same motion pattern once translated to the same origin (see Figure 5.6).

— In the medium case, salient trajectories share either the entrance or the exit with normal trajectories. The other gate of salient trajectories is chosen two positions after or before the other gate of normal trajectories. For instance, if a normal trajectory goes from gates 12 to 9, a salient trajectory may go from gates 12 to 5, gate 5 being two positions after gate 9.

— The most difficult case is built similarly as the medium case. The difference is that the gate of a salient trajectory that differs from the gate of normal trajectories is only one position apart. For instance, for a normal trajectory going from gates 12 to 9, a salient trajectory may go from gates 12 to 7.

In the following, these cases will be designated with their degree of saliency respectively qualified as high, middle and low.

In addition to the saliency degree, we will consider different ratios of salient versus normal

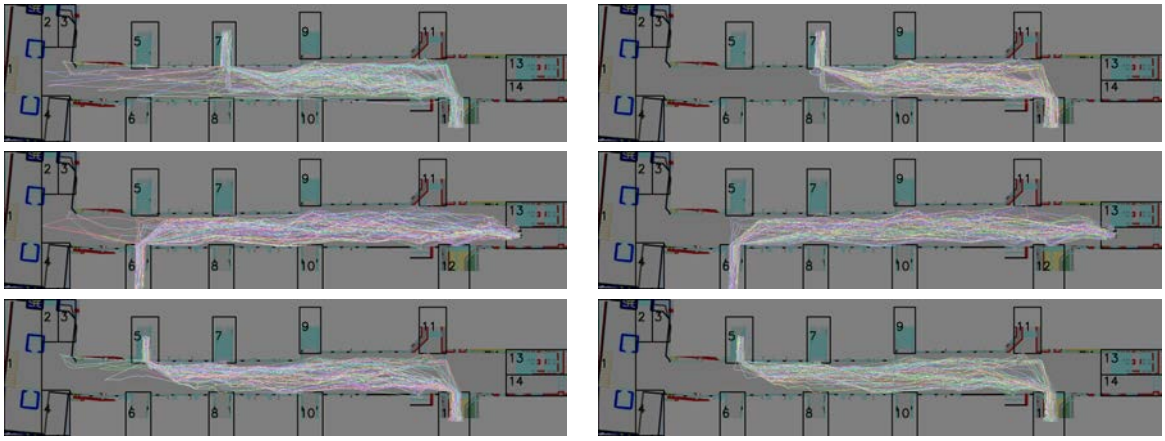


FIGURE 5.6 – Illustration of the RS dataset (corridor PIW, numbers correspond to gates) and of the pre-processing step used to get homogeneous motion patterns. On the left, trajectories that have not been checked yet are displayed. On the right, trajectories that are kept after the pre-processing step are displayed. The trajectories that do not directly go from the entrance to the exit but that make a detour are a minority for these three settings. The post-processing step discards such trajectories.

trajectories. Ratios of 5%, 10%, 15% and 20% will be tested.

5.5.1.2 Evaluation setting RSE-B

The second evaluation setting RSE-B is built for comparison with existing methods. Some of these methods need to be trained on normal data exclusively, and a training set is required for every scenario. The setting RSE-B involves only two entrance/exit pairs, instead of 45 for the RSE-A setting. The normal trajectories are checked by hand to make sure they all follow a consistent motion pattern. This requirement explains that only two types of entrances/exit pairs are included. Salient trajectories are associated to different entrance/exit pairs taken at random as in the easiest case of RSE-A.

200 normal trajectories are selected coming from gate 12 and going to gate 7. We name them as the G12-7 subset. 200 other trajectories are selected coming from gate 12 and going to gate 8. They are designated as the G12-8 subset. Samples are depicted in Figure 5.7.

Then, a group of 100 salient trajectories, relatively to both the G12-7 and G12-8 subsets, is constructed. The G12-7 and G12-8 sets are each divided into a training set of 100 trajectories, and a test set of 100 trajectories. We will call these sets respectively TrG and TeG. The training set TrG is only used for a few methods of the literature [RB18 ; RB19] in the comparative experiments. To get saliency ratios of 5%, the 100 salient trajectories are split into 20 groups of 5 trajectories each. For each G12-7 and G12-8 test subsets, 20 scenarios are built by adding in turn the 20 groups of salient trajectories to the 100 normal trajectories.

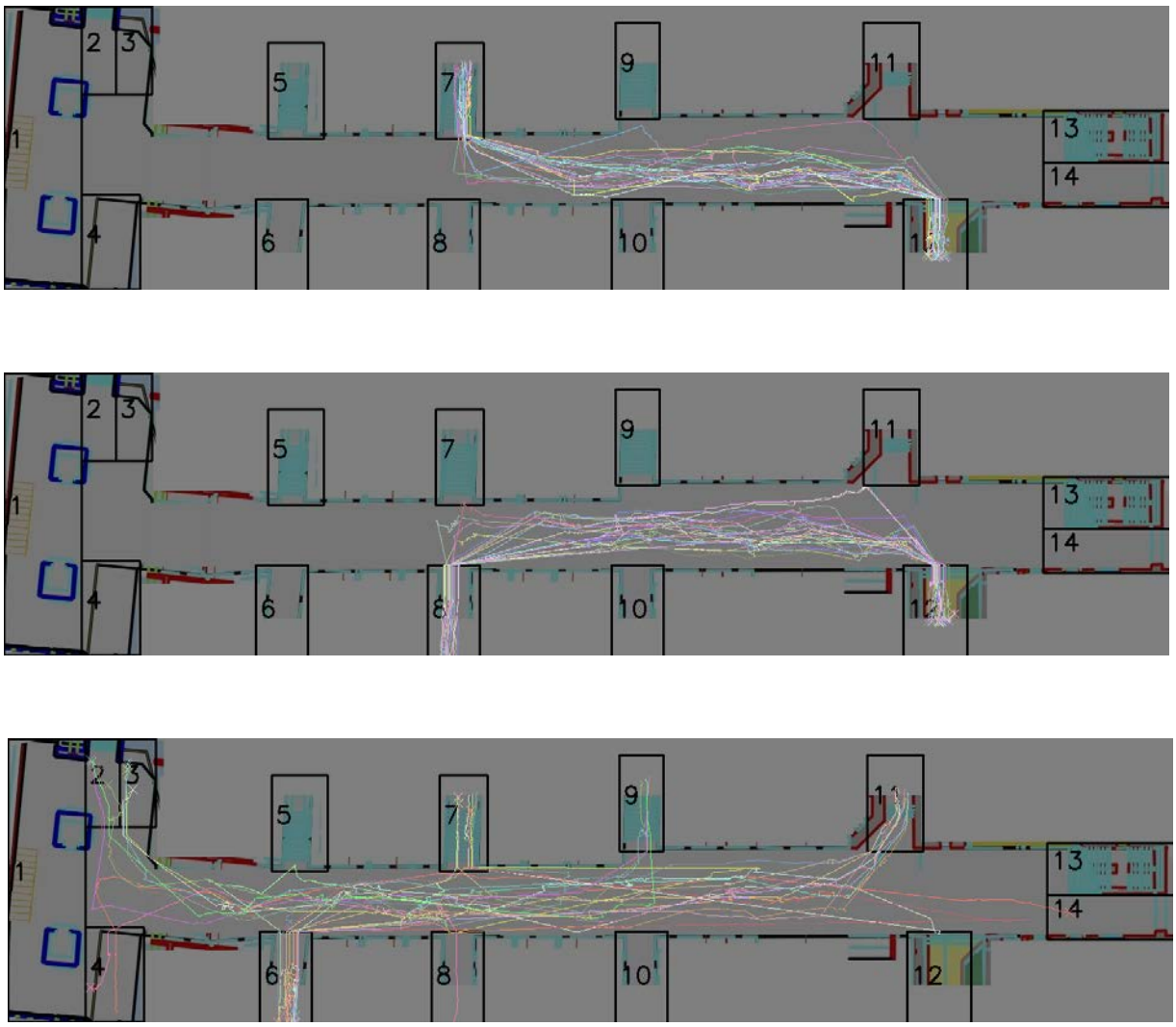


FIGURE 5.7 – From top to bottom : 25 trajectories from the G12-7 subset, 25 trajectories from the G12-8 subset, and 25 salient trajectories with different entrance/exit pairs corresponding to the evaluation setting RSE-B.

5.5.1.3 Evaluation setting RSE-C

Finally, the evaluation setting RSE-C leverages the presence of different motion patterns for a given entrance/exit pair. More specifically, the trajectories that go straight from the entrance to the exit will correspond to the normal trajectories, and all the other ones will correspond to the salient trajectories (see Figure 5.8 for an illustration). To be sure that all trajectories are properly assigned to the normal or salient class in the ground truth, the labelling is performed by hand for this setting. RSE-C will include a total of 155 trajectories going from gate 12 to gate 7, and among them, 137 are normal and 18 are salient.

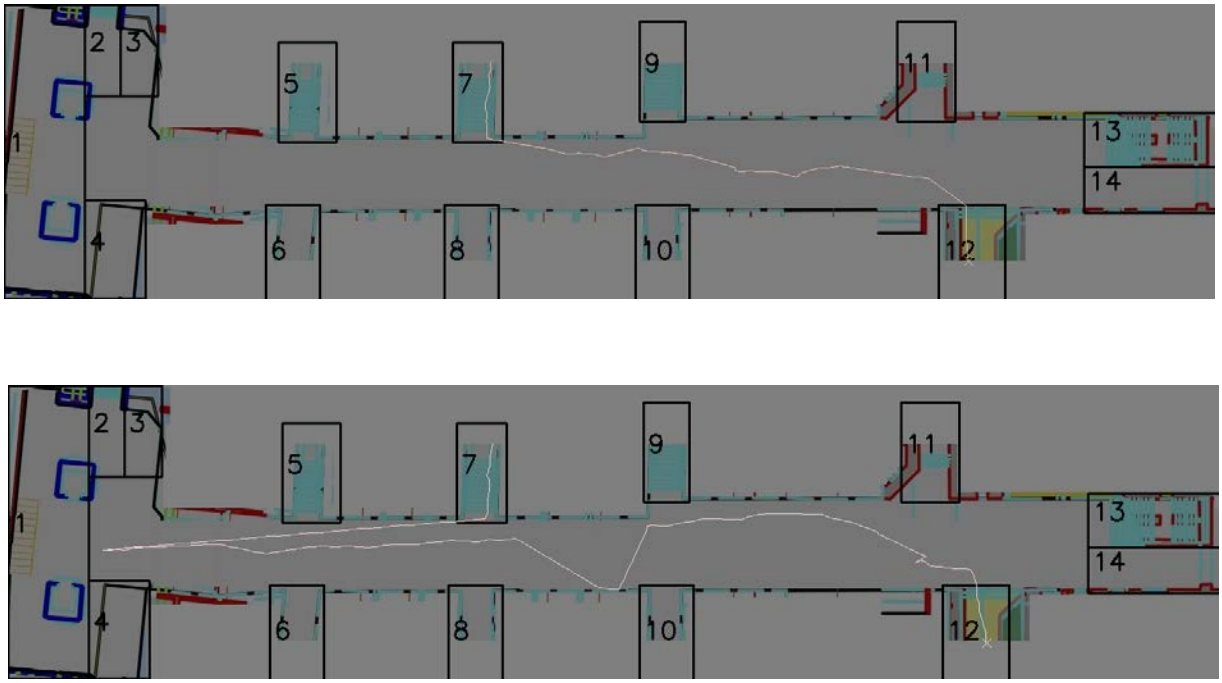


FIGURE 5.8 – A normal trajectory (top) and a salient trajectory (bottom) for the RSE-C evaluation setting.

5.5.2 Network setting

The network was implemented with the PyTorch framework [Pas+19]. For training, we used the Adam algorithm [KB14]. We set the learning rate by training the auto-encoder network on the STMS dataset without adding the consistency constraint in the loss. This was done to find a relevant value for the learning rate that would ensure both a stable training and a good convergence. Indeed, a too high learning rate tends to make the learning unstable, and a too low learning rate tends to make the convergence slow. We did not include the consistency constraint at this stage because, by imposing a second term in the loss potentially conflicting

with the reconstruction term, it might *a priori* make the training harder, thus making more difficult to find a relevant value for the learning rate. A learning rate of 10^{-4} provided a good compromise between speed and convergence and was then retained.

After having set the learning rate, the parameter β of the loss function (eq.5.3), that defines the balance between the reconstruction and consistency constraints, was set as follows. On one hand, a too low value of β would make the consistency constraint negligible, and only the trajectory reconstruction part of the network would survive. On the other hand, a too high value of β would prevent the auto-encoder from correctly reconstructing trajectories. In practice, we set $\beta = 10^5$ for the experiments on the STMS dataset. For the RS dataset, we considered values of $\beta = 10^5$, $\beta = 10^3$ and $\beta = 0$, as mentioned in Section 5.4.2.

Regarding the computation time, the forward pass takes 0.1 second and the backward pass 0.5 second, for a batch of 10 trajectories comprising 40 positions, on a machine with 4 CPU cores of 2.3 GHz. We let each network variant train for around three weeks. We note that we implemented our method (both the network and the loss) by relying heavily on the fact that any valid Python code is a valid PyTorch code, which is possible due to the way backpropagation is handled in this framework. This allows us to reduce the time needed to code the methods, at the expense of the training time. This seemed *a priori* a relevant choice given the relatively low complexity of trajectories compared to image data. Still, a more carefully optimised code would be desirable. We leave this as an open perspective.

5.6 Experimental results

In this section, we report the experiments we carried out on the synthetic STMS dataset and the train station RS dataset.

5.6.1 Synthetic dataset and ablation study

We first evaluate our trajectory saliency method on the STMS synthetic dataset presented in Section 5.4.1. Our first goal is to assess the quality of the trajectory reconstruction by the auto-encoder. The quality score r will be defined as the average reconstruction error $\bar{e}$, divided by the average displacement over the whole dataset $\bar{v}_n$:

$$r = \frac{\bar{e}}{\bar{v}_n}. \tag{5.7}$$

r is defined this way to be dimensionless and comparable between different settings.

Our network is trained on the STMS dataset with the procedure described in Section 5.5. For this dataset, we consider only one variant with β set to 10^5. We got a score $r = 0.62$ for this

dataset. We also visualise samples of reconstructed trajectories. They were accurate enough, as plotted in Figure 5.9.

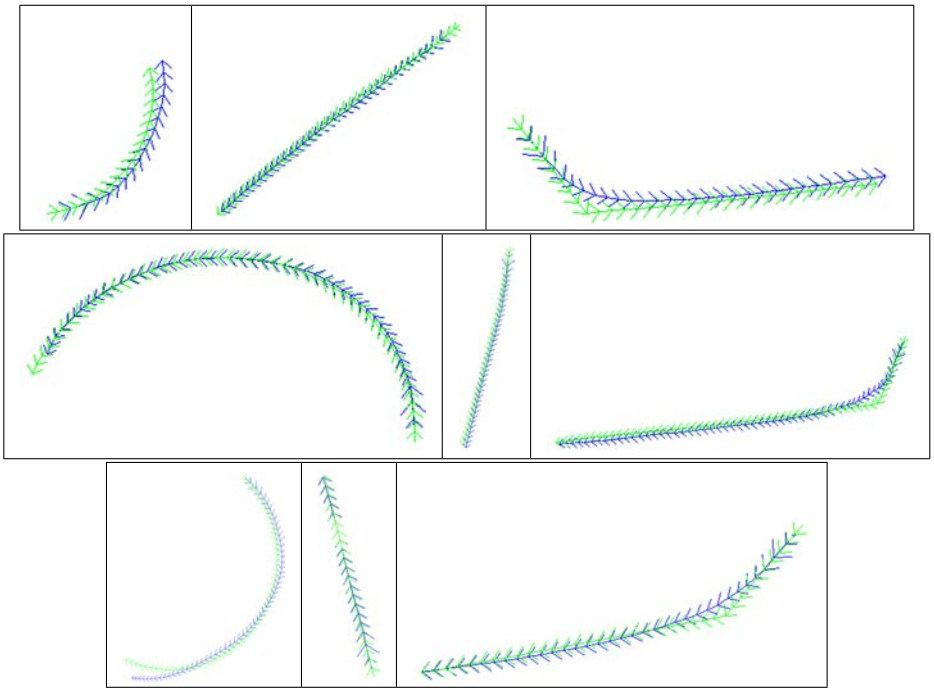


FIGURE 5.9 – Nine examples of synthetic trajectories (in green) and their reconstructed counterpart superimposed (in blue) for the three classes of trajectories. They nicely fit, demonstrating an accurate reconstruction.

The second goal is to assess the trajectory saliency detection performance. We computed precision, recall and F-measure. The latter is the harmonic mean of the two former quantities :

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5.8}$$

The STMS dataset was constructed so that there is on average one salient trajectory every 40 normal trajectories. As shown in Table 5.1, we reached a F-measure equal to 0.89 for this dataset. It proves that we were able to correctly find even subtle trajectory saliency.

To highlight the contribution of the consistency constraint, we conducted an ablation study of our method. More specifically, the consistency constraint is removed, letting the auto-encoder reconstruction constraint drive the latent code estimation. The reconstruction score is $r = 0.58$, and as expected it is slightly better than the one $r = 0.62$ obtained with the additional consistency constraint. However, the F-measure collapses to 0.26 as shown in Table 5.1. This clearly demonstrates the importance of the consistency constraint. Our intuition is that without the consistency constraint, the network employs all the available degrees of freedom to encode the trajectory pattern including any small random noise. With the consistency constraint, the

network is more focused on correctly representing the overall motion pattern of each trajectory, which allows us to better distinguish normal and salient trajectories.

	Precision	Recall	F-measure	λ
Our full method	0.91	0.87	0.89	3.1
Without consistency	0.22	0.31	0.26	2

TABLE 5.1 – Performance of our method evaluated on the STMS dataset, with and without the consistency constraint. We also indicate the decision threshold λ.

5.6.2 Comparative results on the RS dataset

We will now present results on the three settings RSE-A, RSE-B and RSE-C involving real pedestrian trajectories of the train station RS dataset.

5.6.2.1 Results on the evaluation setting RSE-A

Results on the RSE-A setting are collected in Table 5.2, for the four variants of our method, $V\beta_5$, $V\beta_3$, $V\beta_0$ and V_δ, described in more details in Section 5.4.2. Here, let us simply recall that $V\beta_5$, $V\beta_3$ and $V\beta_0$ are the variants for which the parameter on the consistency constraint is respectively 10^5, 10^3 and 0. V_δ is a variant with no consistency constraint, but with more diverse training data. First of all, we observe as expected that the more salient trajectories depart from normal ones, the better the saliency estimation results are. Among $V\beta_5$, $V\beta_3$ and $V\beta_0$, the best method for a given degree of saliency is $V\beta_5$ by a clear margin, which confirms the interest of the consistency constraint. Nevertheless, the V_δ variant offers better performance than $V\beta_0$ by a large margin, and it is somewhat on par with $V\beta_5$. It demonstrates that significant performance improvements can be obtained with a larger dataset and more diverse batches for training.

We now report a second experiment, which consists in varying the ratio of saliency in the test dataset (defined as the ratio between the number of salient and normal trajectories). Let us recall that for the procedure described in Section 5.3.2 to set λ, we built a validation dataset with the same saliency ratio than the test set. Results are collected in Table 5.3 for the best two variants $V\beta_5$ and V_δ. We observe that the overall performance evaluated with the F-measure does not vary much when the ratio of saliency increases, which demonstrates that our method is applicable for different saliency regimes.

5.6.2.2 Results on the evaluation setting RSE-B

We use the saliency evaluation setup presented in Section 5.5.1.2 to compare our method with several existing methods : DAE [RB18], ALREC [RB19] and DRL [Yao+17]. Our method variants are trained as described in Section 5.4.2. The decision threshold λ is set as for the

Method variant	Saliency degree	Precision	Recall	F-measure	λ
$V\beta_5$	Low	<u>0.59</u>	**0.75**	<u>0.66</u>	1.9
$V\beta_3$	Low	0.46	0.57	0.51	2.0
$V\beta_0$	Low	0.49	0.63	0.55	1.9
V_δ	Low	**0.72**	<u>0.69</u>	**0.70**	2.3
$V\beta_5$	Medium	**0.99**	0.86	<u>0.92</u>	3.2
$V\beta_3$	Medium	0.77	<u>0.89</u>	0.82	2.1
$V\beta_0$	Medium	0.85	0.85	0.85	2.4
V_δ	Medium	<u>0.96</u>	**0.93**	**0.94**	2.8
$V\beta_5$	High	**0.96**	**1.0**	**0.98**	1.6
$V\beta_3$	High	0.94	0.99	0.96	1.8
$V\beta_0$	High	0.92	**1.0**	0.96	1.8
V_δ	High	<u>0.95</u>	**1.0**	<u>0.97</u>	1.5

TABLE 5.2 – Evaluation setting RSE-A of our four method variants for a saliency ratio of 5%. For each level of saliency degree, the best performance is in bold, the second best is underlined.

evaluation setting RSE-A, over a validation set with low saliency degree and saliency ratio of 5%.

DAE and ALREC methods estimate saliency by training an auto-encoder on normal data only, and by considering that trajectories poorly reconstructed are salient. Then, for a fair comparison, we trained each of the two networks on the training subset G12-7 and on the training subset G12-8 respectively according to the experiment conducted. We then estimated trajectory saliency first on the G12-7 subset and then on the G12-8 test subset. The training subsets include 100 trajectories per scenario, which is more than the 16 to 31 trajectories per scenario that the authors considered in their own work. Due to the higher number of trajectories, we did not include data augmentation.

The DRL method represents trajectories with a constant-length code. This representation is obtained directly by training a network on the test trajectories. DRL was designed to produce input to clustering algorithms. To apply DRL to the trajectory saliency detection task, we considered two cases. The first option consists in using the *k*-means clustering algorithm with two classes. We then state that the class with fewer elements is the salient class. However, there is a risk that the large imbalance between the normal and salient classes makes this first option unstable. It motivates the following alternative. The second option exploits a similar decision procedure than ours but applied to the code produced by DRL. We took 101 values of λ regularly sampled in $[0, 5]$, and we selected the one that provided the best performance for DRL on the test set.

Results on the two test subsets are given in Table 5.4. We can observe that our method yields the best performance as expressed by the F-measure for both the G12-8 and G12-7

Method variant	Saliency visibility	Saliency ratio	Precision	Recall	F-measure
$V\beta_5$	Low	5%	0.59	0.75	0.66
V_δ	Low	5%	0.72	0.69	0.70
$V\beta_5$	Low	10%	0.75	0.67	0.71
V_δ	Low	10%	0.79	0.64	0.71
$V\beta_5$	Low	15%	0.82	0.63	0.72
V_δ	Low	15%	0.74	0.75	0.74
$V\beta_5$	Low	20%	0.78	0.63	0.70
V_δ	Low	20%	0.83	0.58	0.68
$V\beta_5$	Medium	5%	0.99	0.86	0.92
V_δ	Medium	5%	0.96	0.93	0.94
$V\beta_5$	Medium	10%	0.92	0.98	0.95
V_δ	Medium	10%	0.92	0.99	0.95
$V\beta_5$	Medium	15%	0.95	0.98	0.96
V_δ	Medium	15%	0.91	0.99	0.95
$V\beta_5$	Medium	20%	0.91	0.98	0.94
V_δ	Medium	20%	0.94	0.98	0.96
$V\beta_5$	High	5%	0.96	1.0	0.98
V_δ	High	5%	0.95	1.0	0.97
$V\beta_5$	High	10%	0.92	1.0	0.96
V_δ	High	10%	0.97	1.0	0.98
$V\beta_5$	High	15%	0.91	0.98	0.94
V_δ	High	15%	0.97	0.98	0.97
$V\beta_5$	High	20%	0.93	0.95	0.94
V_δ	High	20%	0.98	0.95	0.96

TABLE 5.3 – Evaluation setting RSE-A, with different saliency ratios for the best two performing variants our method.

datasets. It even provides the best precision and recall scores in all cases.

RS subset	G12-7			G12-8		
Method	P	R	FM	P	R	FM
DAE [RB18]	0.83	0.32	0.46	0.81	0.32	0.46
ALREC [RB19]	0.72	0.43	0.54	0.71	0.45	0.55
DRL-KMean [Yao+17]	0.05	0.54	0.10	0.08	0.53	0.14
DRL-C [Yao+17]	0.72	0.24	0.36	0.54	0.52	0.53
Ours $V\beta_5$	**1.0**	0.98	**0.99**	**1.0**	0.89	0.94
Ours $V\beta_3$	**1.0**	0.98	**0.99**	**1.0**	0.98	**0.99**
Ours $V\beta_0$	**1.0**	**0.99**	**0.99**	**1.0**	**0.99**	**0.99**
Ours V_δ	**1.0**	0.95	0.97	**1.0**	0.93	0.96

TABLE 5.4 – Comparative results for the RSE-B evaluation setting, on the G12-8 and G12-7 test subsets. P, R and FM denote respectively the precision, recall and F-measure. Best results are in bold.

In addition to this objective experimental comparison, we now discuss inherent differences between our method and methods such as DAE and ALREC. DAE and ALREC methods build a model to represent the normal data, and then, expect that this model will fail for anomalous elements. The failure level of the model is interpreted as the trajectory saliency indicator. A major limitation of this paradigm is its low generalisation capability. Indeed, for a new configuration, the normal and abnormal trajectories are likely to change, requiring to train again the network. For a deployment to many different scenarios, this can quickly become unpractical. More fundamentally, the point is that these approaches are by design unable to handle relative saliency. Relative saliency refers to situations where an element is salient only because it is different from its local context. If the context changes the same element may be non salient.

In contrast to this family of methods, our approach is able to handle relative saliency, as illustrated by the experiments presented in Sections 5.6.1 and 5.6.2. Indeed, for these experiments, a trajectory appears as salient in a given context, while in a different one the same trajectory may be non salient. Furthermore, we do not rely on the assumption that a salient trajectory is not present in the training set. In fact, if salient trajectories are included in the training set, we expect they will be properly reconstructed and encoded, thus facilitating the decision. It enables us to train the network only once for each dataset, whatever the scenario considered.

5.6.2.3 Results on the evaluation setting RSE-C

We report results on the RSE-C described in Section 5.5.1.3. The decision threshold λ is set as for the evaluation setting RSE-A, more precisely the case with low saliency degree and a saliency ratio of 15% since more salient trajectories are expected. Results are collected in

Table 5.5 for our method variants $V\beta_5$, $V\beta_3$, $V\beta_0$ and V_δ.

We observe that a higher consistency constraint is here not desirable. In fact, this setting has specific characteristics. The salient trajectories are visually different, but they still have the same entrance and exit than normal trajectories. Accordingly, a compromise between the reconstruction and consistency constraint is likely to restrict the representation of salient trajectories to a more direct path between the entrance and the exit, which is precisely the normal case.

Finally, we observe that for RSE-C, V_δ performance is behind $V\beta_0$ and $V\beta_3$ ones. It suggests that for this particular experimental setting, the batches built for the training stage should not include too diverse trajectories.

Method variant	Precision	Recall	F-measure	λ
$V\beta_5$	0.11	0.06	0.07	1.6
$V\beta_3$	**0.86**	0.67	<u>0.75</u>	1.5
$V\beta_0$	<u>0.82</u>	**0.78**	**0.80**	1.4
V_δ	0.68	<u>0.72</u>	0.70	1.3

TABLE 5.5 – Evaluation setting RSE-C. The best performance is in bold, the second best is underlined.

5.7 Comparison with possible alternatives

In this section, we further analyse two choices that we made in the definition of our method by comparing them to alternatives. First, we investigate other types of descriptor of the degree of saliency. Then, we discuss an alternative approach to set the decision threshold λ.

5.7.1 Other trajectory saliency descriptors

Our trajectory saliency detection relies on the normalized distance between each trajectory code and the median one (as defined in Equation 5.5). It can be viewed as a descriptor of the degree of saliency. In this section, we introduce five descriptors, before comparing them experimentally.

In all cases, normality is represented by the median code $\widetilde{c}$ computed for each scenario. The simplest way to measure the difference between a trajectory code c_i and $\widetilde{c}$ is to compute the Euclidean distance d_i between them. From d_i several descriptors can be defined.

First, d_i itself can be taken as descriptor.

Normal trajectories are expected to be close to the median trajectory, but they are also expected to include minor deviations to it. This means that the d_i coefficients will differ from

zero in practice. To mitigate this, a first option consists in subtracting the typical distance to d_i observed in the scenario. This leads to the two following descriptors r_i and s_i :

$$r_i = |d_i - \widetilde{d}| \tag{5.9}$$

$$s_i = |d_i - \bar{d}| \tag{5.10}$$

where $\widetilde{d}$ is the empirical median of distances d_i and $\bar{d}$ the empirical average of distances d_i. $\widetilde{d}$ and $\bar{d}$ are computed for each scenario. The difference between these two descriptors is that, with the median, the resulting descriptor is less affected by the presence of salient elements.

Before proposing further refinements, let us note that the codes are output of a LSTM layer. The last operation of this layer is a product between a term that has been processed by a sigmoid, and between a term that has been processed by a hyperbolic tangent, which implies that code values lie in $[-1, 1]$. Although this range is limited, there is no strict guarantee that for two different scenarios, the range of normal codes will be identical. A solution is then to divide the descriptor by the expected variation. This leads to the definition of two new descriptors p_i and q_i. q_i is the one used so far as defined in Section 5.3.1. It is recalled here for convenience. We have :

$$p_i = \frac{|d_i - \widetilde{d}|}{\text{MAD}} \tag{5.11}$$

$$q_i = \frac{|d_i - \bar{d}|}{\sigma} \tag{5.12}$$

where σ the standard deviation of the d_i of the scenario, and MAD the median absolute deviation of the d_i of the scenario. The median absolute deviation is defined as MAD $=$ median$(|d_i - \widetilde{d}_i|)$. Similarly as for r_i and s_i, p_i is expected to be more robust to the presence of outliers compared to q_i, due to its median-based definition.

Now that the d_i, r_i, s_i, p_i and q_i descriptors have been defined, we compare them experimentally on the STMS dataset. For this, we apply the saliency detection algorithm presented in Section 5.3.2, except that we replace the q_i coefficient by each alternative. Results are given in Table 5.6. First, we observe that all kinds of normalisation improve the results compared to descriptor d_i. We also observe that both centering and dividing by the expected variation provide improvements.

Surprisingly the mean is more beneficial than the median. This is more obvious when dividing by the expected deviation, either σ or MAD. The median is by construction more robust to the presence of outliers compared to the mean. These results demonstrate that for this component of our saliency detection method, it is not desirable to be robust to the presence of outliers. In fact, there is a simple explanation to this. When the standard deviation is used to

normalise the descriptor, the presence of salient elements in the scenario will lead to a larger σ. This will in turn lower the descriptor value for normal elements, which is finally the desired behaviour. This effect is clearly visible in Figure 5.10. On the other hand, when using the MAD for the normalisation, there is no such effect to facilitate the decision in presence of salient elements.

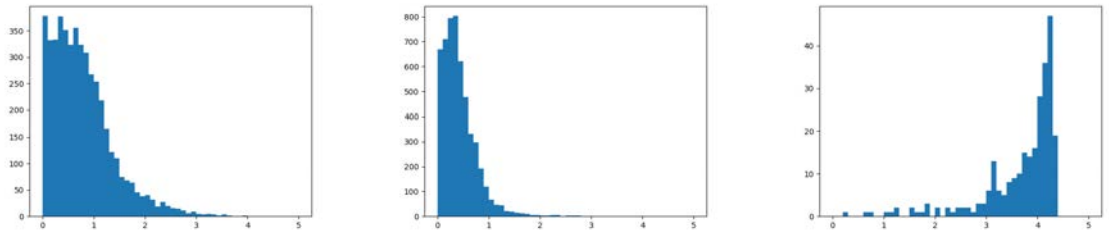


FIGURE 5.10 – Histogram of the non salient q_i coefficients in non salient scenarios (left), histogram of the non salient q_i coefficients in salient scenarios (middle), histogram of the salient q_i coefficients (right). The 500 scenarios of the STMS validation are represented. We can see that in presence of saliency, the q_i associated to non salient trajectories have lower values. This helps distinguishing between salient and normal elements.

These results confirm that the q_i descriptor is more relevant, since it outperforms the four other descriptors by a large margin. This justifies its adoption.

Descriptor	Normalisation	Precision	Recall	F-measure
d_i	None	0.66	0.69	0.67
r_i	Median-based	0.78	0.69	0.73
s_i	Mean-based	0.78	0.77	0.77
p_i	Median-based	0.79	0.81	0.77
q_i	Mean-based	**0.90**	**0.88**	**0.89**

TABLE 5.6 – Comparison of five descriptors to estimate trajectory saliency, on the STMS test set. Best results are in bold.

5.7.2 Alternative method to set the decision threshold λ

In addition to the procedure described in Section 5.3.2 to set the threshold λ, we considered an alternative based on the *p*-value scheme. In this section, we first describe this approach, before discussing experimental observations.

5.7.2.1 Description of the approach

The *p*-value scheme aims to statistically control the number of false detections and enables to automatically set λ. We assume that the q_i descriptors for normal trajectories follow a known probability distribution. We then need to estimate its parameters to fit the empirical distribution of the q_i values of a representative set of normal trajectories. In order to identify candidate distributions, we looked at several histograms of q_i descriptors and observed that their distribution is skewed. We then identified three relevant candidates that we tested on our data. The tested probability distributions were the Weibull distribution, the Dagum distribution in the standardized form and the Dagum distribution in the general form. The expression of each law is the following :

1. Weibull distribution :
$$f(x) = \frac{c}{\gamma}\left(\frac{x}{\gamma}\right)^{c-1} \exp\left(-\left(\frac{x}{\gamma}\right)^{c}\right) \tag{5.13}$$
with $c > 0$ the shape parameter and $\gamma > 0$ the scale parameter of the law.
2. Dagum distribution, standardised form :
$$f(x) = \frac{kx^{k-1}}{(1+x^s)^{1+\frac{k}{s}}} \tag{5.14}$$
with $k > 0$ and $s > 0$ the shape parameters of the law and $x \in \mathbb{R}^+$.
3. Dagum distribution, general form :
$$f(x) = \frac{k}{\gamma}\frac{\left(\frac{x}{\gamma}\right)^{k-1}}{\left(1+\left(\frac{x}{\gamma}\right)^{s}\right)^{1+\frac{k}{s}}} \tag{5.15}$$
with $k > 0$ and $s > 0$ the shape parameters of the law, $\gamma > 0$ the scale parameters of the law and $x \in \mathbb{R}^+$.

The empirical and estimated probability distributions are plotted in Figure 5.11 for the STMS validation dataset. With this dataset, 10000 q_i corresponding to normal trajectories are computed. The laws are fitted with the maximum likelihood estimate method. The estimated values of their respective parameters are $c = 1.19$ and $\gamma = 0.62$ for the Weibull distribution, $k = 0.94$, $s = 3.54$ for the standardised form of the Dagum distribution, and $k = 1.04$, $s = 3.14$ and $\gamma = 0.87$ for the general form of the Dagum distribution.

Visually, the best fit is obtained with the general form of the Dagum distribution. In addition, to get a quantitative measure of the fitting error, we use the criterion $\mathcal{F}$ defined as follows :

$$\mathcal{F} = \sum_b |\mathcal{G}(b) - \mathcal{H}(b)| \tag{5.16}$$

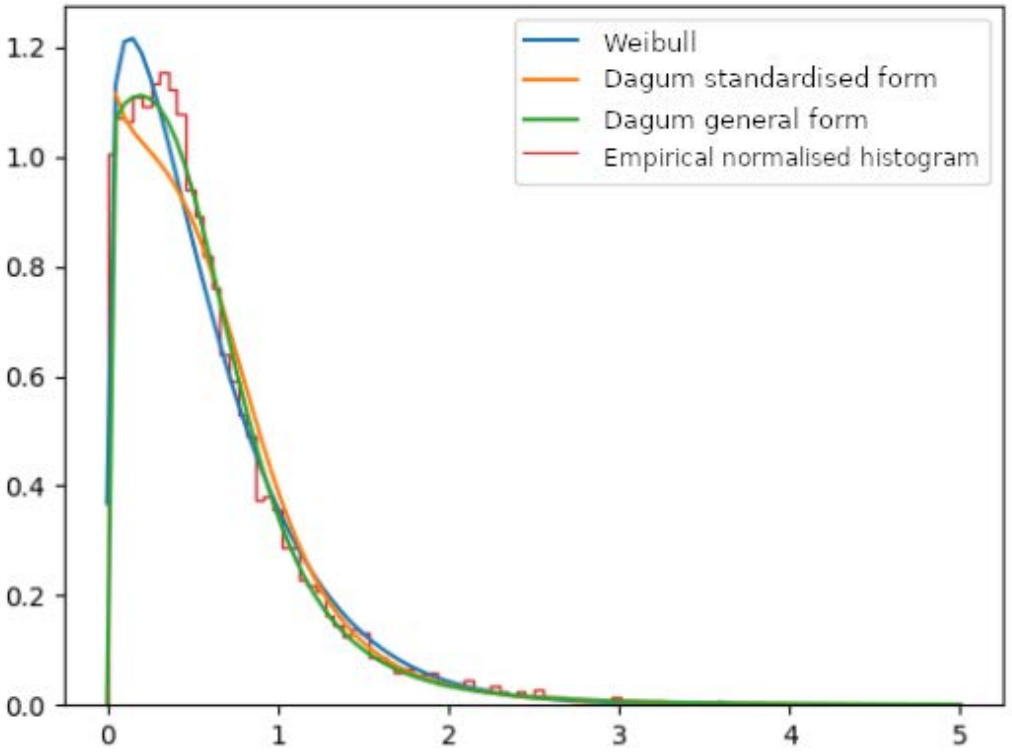


FIGURE 5.11 – Plots of the empirical distribution of the q_i descriptors and the fitted distributions. They are computed from the 10000 normal trajectories of the STMS validation dataset. Each scenario is processed separately to compute the q_i descriptors.

with b a bin, $\mathcal{G}(b)$ the area under the curve of the fitted probability law for the bin b, and $\mathcal{H}(b)$ the observed proportion of elements falling into the bin b. In practice, we use 101 bins regularly sampled between 0 and 5.

The fitting errors are 0.05, 0.10 and 0.08 respectively for the Dagum distribution with the general from, the Dagum distribution with the standardised form and the Weibull distribution respectively as indicated in Table 5.7. This confirms the visual evaluation. Consequently, we adopt the Dagum distribution with the general form to model the data.

Law	Weibull	Dagum standardised	Dagum general
Fitting error	0.08	0.10	0.05

TABLE 5.7 – Fitting errors for the three distributions.

Once the three parameters of the general Dagum distribution are estimated, we can fix the *p*-value and then get the threshold value λ. The *p*-value is supposed to control the proportion of false positives. In an ideal case, the estimated distribution should correspond also to the test data. At this point, let us emphasize that we have no guarantee that this is the case. Indeed, the q_i descriptors depend on the presence of outliers, that is, salient trajectories, through the normalisation with $\bar{d}$ and σ the computed mean and standard deviation. In particular, the standard deviation σ may be noticeably influenced by salient trajectories, despite their rarity. The experiments reported in Section 5.7.1 demonstrated that the q_i descriptors were the best choice for trajectory saliency detection. However, it also means that the fitted probability law may become inappropriate in presence of outliers. Still, since the normalisation with σ tends to make the q_i smaller, it means that fewer q_i will be above the threshold λ compared to the same scenario with no saliency. The *p*-value does not correspond any more to the proportion of false positive, but provides an upper bound for this quantity. If this upper bound is not too large, the *p*-value scheme could prove useful.

5.7.2.2 Experimental observations

To test the *p*-value scheme, we take the evaluation setting RSE-A with the most difficult saliency setting. Because this setting is the most difficult, it is expected to better show the behaviour of the method, where it succeeds and where it fails. To fit the distribution, we need a dataset with only normal trajectories. The TrG dataset has been built manually to meet this criterion, and we use it to fit the law. We consider the V_δ variant of the network, which performed best on the RSE-A setting. Results are given in Table 5.8 for saliency ratios from 0.05 to 0.20. The chosen *p*-value and the False Positive Rate (FPR), which corresponds to the ratio between the number of false positives and the number of normal trajectories, are given. The precision, recall and F-measure are also evaluated.

Before commenting the results, let us recall the goal of the *p*-value approach. The *p*-value approach converts the problem of setting the value of λ into the problem of choosing the proportion of false positive in the prediction. λ is not directly interpretable, contrarily to the proportion of false positive, which makes the choice of a relevant value easier with the *p*-value scheme. However, the results confirm that the empirical probability law changes in presence of outliers for the q_i coefficients. We can see it because the *p*-value and the FPR, which should normally have similar values (the FPR can be viewed as the empirical *p*-value), differ largely in practice. Then, the *p*-value scheme no longer provides a way to set the threshold λ by setting an interpretable value, consequently loosing its interest.

A way to solve this difficulty would be to define alternatives to the q_i coefficients that would not depend on the presence of saliency. This is what was done in Section 5.7.1. The experiments explicitly showed that the variants investigated which removed this dependency were not able to discriminate as well normal and salient trajectories, so we did not retain them. Another possibility would be to retain the q_i coefficients, but to compute $\bar{d}$ and σ once and for all for normal trajectories, obtained on a validation set for each scenario. This way, the dependence on the presence of saliency at test time disappears. However, tests we conducted on the RS dataset in this direction showed the results obtained this way are not as good as the ones obtained with the standard q_i. Similarly to what happened in Section 5.7.1, a possibility is that removing the dependency on the presence of salient elements lowered the results. There is a second possibility that cannot be ruled out. In order to get the normal trajectories on the RS dataset for each scenario, we resorted to the pre-processing step we used previously to ensure all trajectories of a pack are similar. This pre-processing step keeps the trajectories whose length is similar to the median length. However, we have no guarantee that the trajectories of the validation will be identical to the trajectories of the test for a given scenario. For instance, most trajectories in the test may go directly from the entrance to the exit, while most trajectories in the validation may make a detour. In such a case, the descriptors computed for the validation would no longer correspond to the test scenario, possibly deteriorating the results. Anyway, this shows that the requirement of this approach may not be easy to validate in practice. In the end, we then did not retain this approach.

5.8 Additional experiments and visualisations

In this section, we present additional experiments to better understand the behaviour of our method, and to provide more details on a few specific issues.

Saliency ratio	*p*-value	λ	FPR	Precision	Recall	F-measure
5%	0.025	2.4	0.011	0.73	0.67	0.70
5%	0.05	1.9	0.018	0.64	0.76	0.70
10%	0.05	1.9	0.012	0.82	0.61	0.70
10%	0.10	1.5	0.026	0.72	0.74	0.73
15%	0.075	1.7	0.014	0.86	0.61	0.72
15%	0.15	1.3	0.036	0.75	0.76	0.75
20%	0.10	1.5	0.015	0.87	0.54	0.67
20%	0.20	1.2	0.050	0.72	0.68	0.70

TABLE 5.8 – Evaluation setting RSE-A, and λ set with the *p*-value scheme. FPR denotes the False Positive Rate.

5.8.1 Sensibility to the choice of λ

To assess the sensibility of the method to the value of threshold λ, the trajectory saliency detection was conducted on the STMS validation set with a regular sampling of λ (every 0.05) in the interval $[0, 5]$. The corresponding precision, recall and F-measures are plotted in Figure 5.12. It shows that in this case there is a plateau around $\lambda = 3$, which means that the performance is robust to variations of the decision threshold.

5.8.2 Initial state of the LSTMs

The network designed for trajectory representation includes LSTM layers in the encoder and the decoder. In a recurrent network, the value predicted for the next time step is estimated with the help of a hidden state that keeps memory of past information. However, there is no past information for the first time step. In practice, there are several ways to initialise the hidden LSTM state (see for instance [ZTG12]). The simplest solution consists in setting the initial state to a constant value, by default to 0. Slightly more sophisticated possibilities consist in setting it randomly or in learning it. If the initial value is randomly chosen, it is a way to introduce data augmentation, since the network faces more diverse configurations for the same training dataset. It may consequently help to improve performance. Learning the initial state can be done by including the initial state as a variable in the backpropagation algorithm. In preliminary experiments, we did not find any significant difference between these different initialisations in the network performance. We then decided to adopt a random initialisation, by drawing the initial state from a normal distribution. This way, we get a data augmentation step at no extra cost, which may be beneficial.

With a random initialisation of the hidden state, the resulting parametrization of the network is of course not strictly deterministic. Still, the network delivers consistent predictions when the

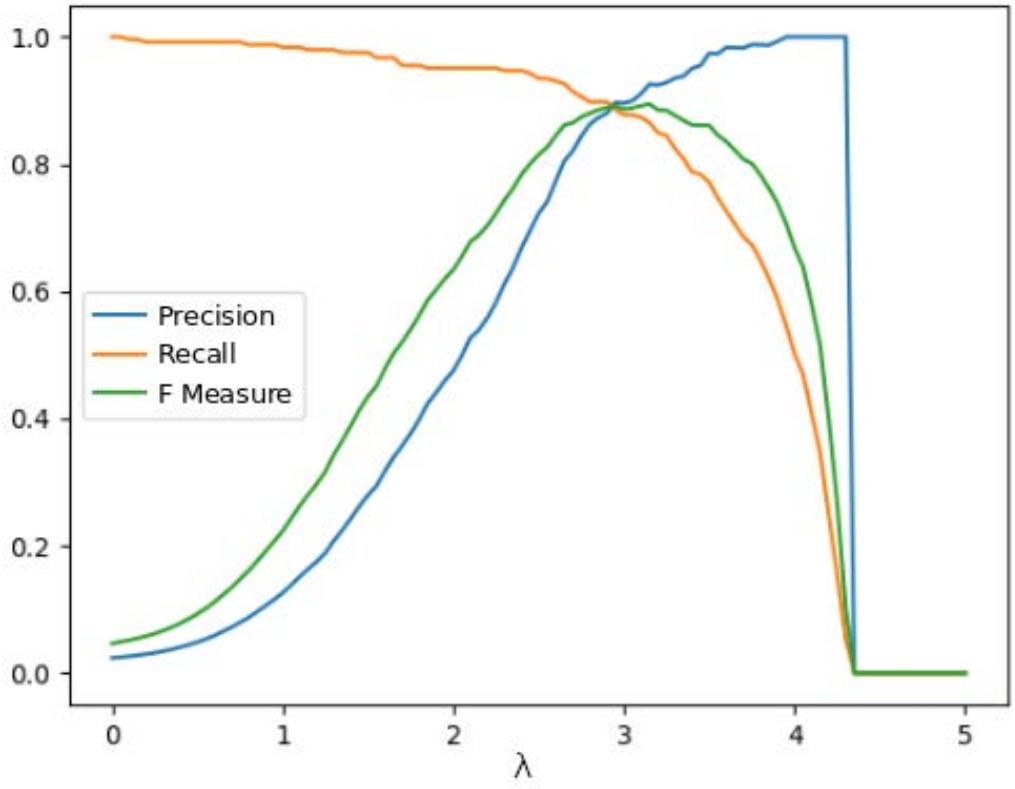


FIGURE 5.12 – Precision, recall and F-measure quantifying the trajectory saliency detection performance for different values of λ on the STMS validation set. λ values are sampled every 0.05 within [0,5].

same trajectory is given as input several times. This is illustrated in Figure 5.13 displaying several reconstructions of the same trajectory for several random initialisations. Apart from a hardly visible variation near the starting position of the trajectory (bottom-right), the reconstructions are almost identical.

5.8.3 Evolution of the model during training

Our method builds constrained latent codes to represent trajectories, and the codes are exploited to detect trajectory saliency. Our decision framework takes benefit from the location of codes in the latent space. It is somewhat inspired by deep metric learning. For the successful training of a method of this family, we have to be aware of a possible behaviour described below.

Figure 5.14 displays the evolution of the trajectory reconstruction error and the F-measure related to saliency detection on the STMS validation set along the training iterations. Each iteration corresponds to a batch of 60 to 66 trajectories, depending on the random inclusion of salient trajectories. When looking first at the reconstruction performance, we observe that, after an initial phase with large improvements, the gain in performance becomes slower and slower. There is a noticeable improvement around iteration 200000, even if the final gain remains limited. On the other hand, regarding saliency detection, performance reaches a plateau in a few iterations, and even deteriorates a little for a while. Only after more than 100000 iterations, performance improves again relatively quickly, before reaching another plateau.

A similar behaviour is observed for deep metric learning [HBL17]. Indeed, codes are expected to evolve in the latent space during learning, in order to correctly represent different elements into separate clusters. However, while the clusters are not clearly distinguishable, the saliency detection algorithm cannot work well. As a consequence, it is beneficial to let the training go on, even if performance apparently stagnates.

5.8.4 Alternatives investigated for the RS dataset

The trajectories of the RS dataset are more irregular and more challenging to reconstruct compared to the STMS dataset. Reaching a sufficient precision for the trajectory reconstruction task is a pre-requisite for the trajectory saliency detection method. We present variants we investigated to try to improve the performance of the auto-encoder. They correspond to some options or hyperparameters of the network and its training.

The first option concerns the training modality of the network : from scratch on the RS dataset, or pre-training first on the STMS dataset. Although the two datasets are not similar, pre-training the network may allow it to learn useful features.

Data pre-processing is another common step to improve performance. As explained in Section 5.4.2, we noticed that the trajectories of the RS dataset are often long in term of number

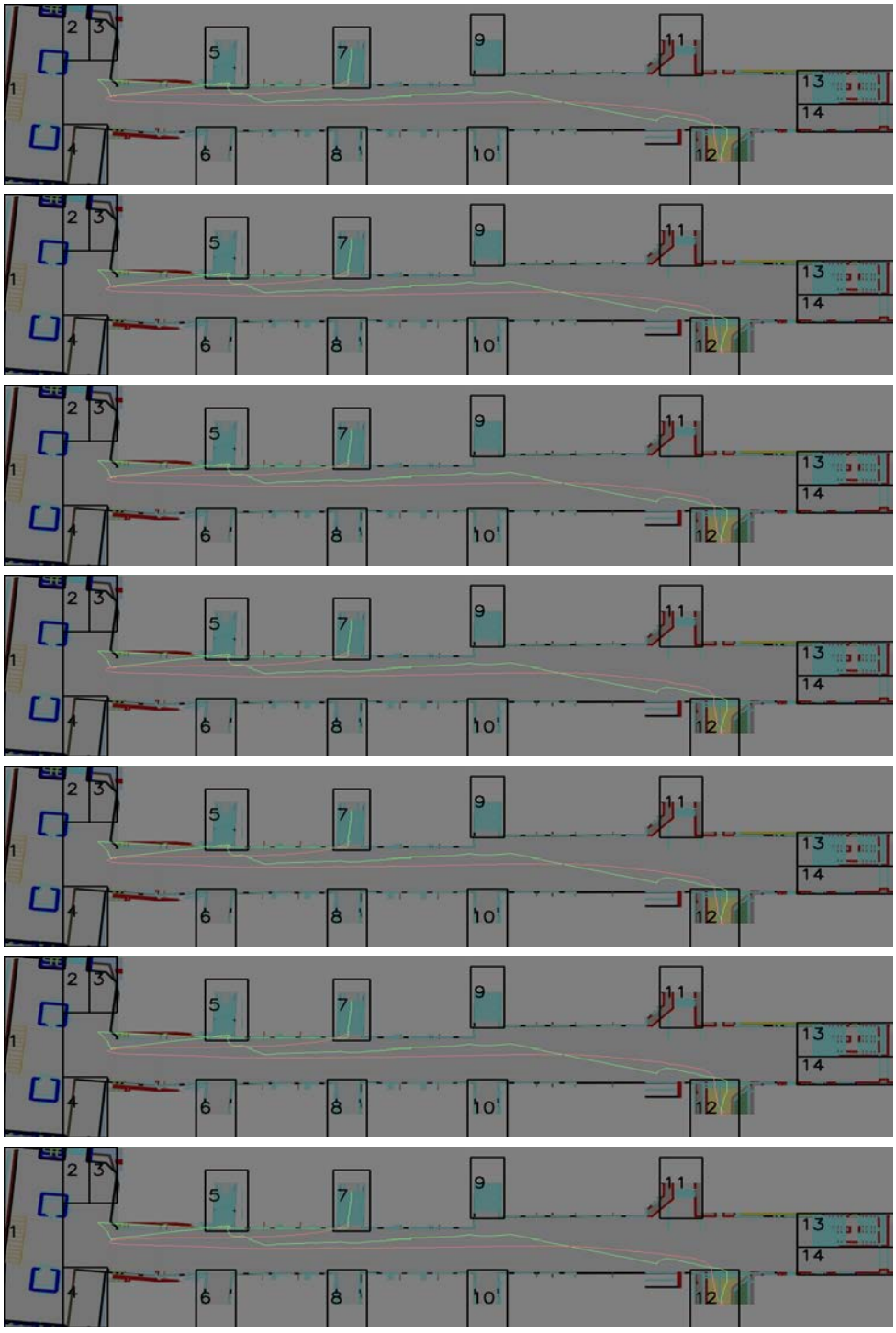


FIGURE 5.13 – One trajectory reconstructed several times by a network with different initialisations of the hidden state of the LSTM. The reconstructed trajectory is drawn in red and the ground truth trajectory is drawn in green. The initial state of the LSTM is random, but the impact on the reconstructed trajectory is minimal.

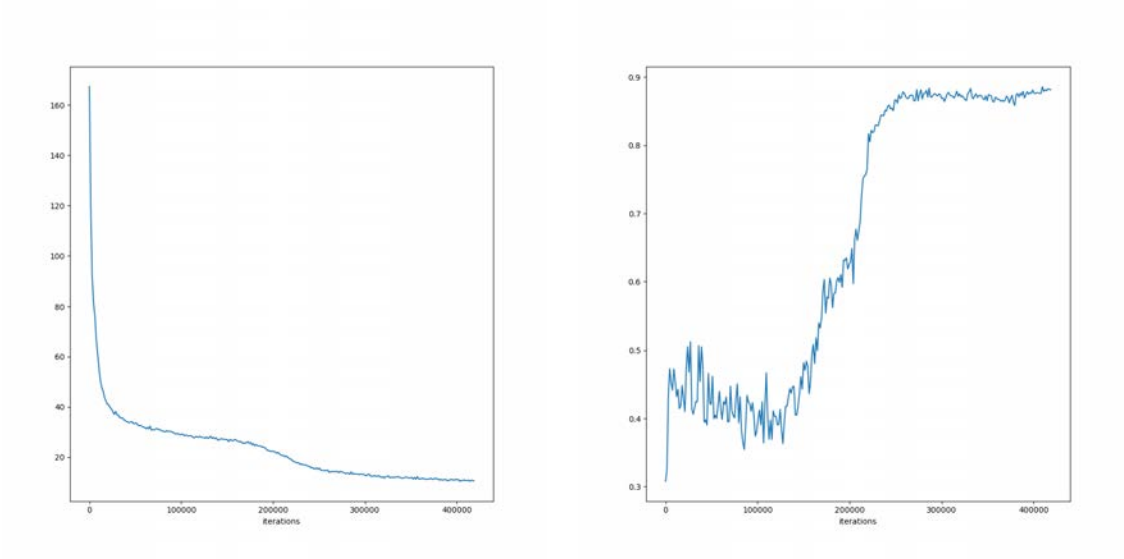


FIGURE 5.14 – Evolution of the trajectory reconstruction error (left) and of the F-measure related to trajectory saliency detection (right) on the validation set of the STMS dataset during training. Each iteration corresponds to a batch of 60 to 66 trajectories (depending on the random inclusion of salient trajectories).

of points involved. They also exhibit higher motion magnitude than STMS trajectories. We then tried to subsample them by taking one every five points. We also considered normalizing the motion-step magnitude, with the objective to make them more similar to the trajectories of the STMS dataset.

Another idea was to remove the (x,y) positions from the input of the encoder, to keep only velocities. This was done to test how the network would behave when fed with less redundant input.

Finally, we attempted to adapt the magnitude of the consistency constraint. The weights tested were $\beta = 10^5$ (same as for STMS), $\beta = 10^3$ and $\beta = 0$. The latter obviously means no consistency constraint. We expect that lowering the consistency constraint will make the reconstruction fit easier, but it will also decrease the enforcement of the consistency on the codes. It means that, looking at the reconstruction accuracy alone will not be sufficient to assess the final impact, it will also be necessary to perform an evaluation of trajectory saliency detection. We refer to Section 5.6 for this. Let us note that the experiments reported below were carried out with an early version of the training protocol regarding the consistency constraint. More specifically, the consistency constraint was applied to batches of trajectories taken at random. The role of the consistency was then limited to a rougher and more generic regularisation of the codes. Nevertheless, the experiments with this earlier version of the consistency constraint allowed us to explore and find relevant ways to train the network. Conducting these experiments

again with the most recent version of the consistency would be of limited interest. Therefore, we present the original experiments here.

Given the number of hyperparameters and the time needed to train the network, we were not able to investigate all the possible combinations. Instead, we only tested the most relevant ones. Our objective was rather to find a good set of hyperparameters allowing us to properly reconstruct the trajectories while being a good setting for trajectory saliency detection.

Table 5.9 collects the reconstruction errors for several combinations of the hyperparameters introduced above, along with the reconstruction error obtained on the STMS dataset. The reconstruction error r is defined as the average reconstruction error divided by the average displacement, both computed for the whole dataset, as defined in Section 5.6.1 in equation 5.7. Subsampling and re-scaling of trajectories are operations that modify the average displacement for the whole dataset, which is taken into account in the computation of r.

The table demonstrates that fine-tuning the network, along with subsampling and rescaling the data, lead to improvements compared to the baseline. On the other hand, removing the position in the input decreases performance, and then was abandoned. Finally, we can observe that lowering the consistency slightly improves the reconstruction performance. However, the reconstruction error is acceptable for the different values of the consistency parameter, and it is of the same order as the reconstruction error obtained for the more regular STMS trajectories.

5.8.5 Additional experiments on a biological trajectory dataset

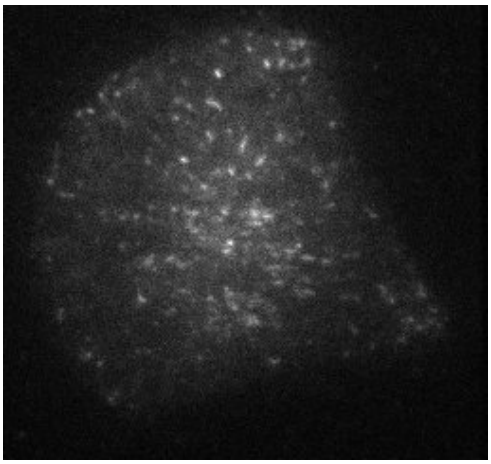


FIGURE 5.15 – 2D light microscopy image of proteins in a cell. 2D Total Internal Reflection Fluorescence (TIRF) microscopy has been used to obtain this image. The acquisition has been made by UMR 144 CNRS, Institut Curie, PICT IBiSA. The tracking of the proteins of interest in the image sequence supplies their trajectories.

We have applied our trajectory saliency detection method on a very different set of trajec-

Configuration	Reconstruction error r
STMS	0.52
RS *c5*	104
RS *c3*	104
RS *c0*	105
RS *c5* + ft	108
RS *c5* + np	106
RS *c5* + sb	19.6
RS *c5* + dv	69
RS dv + *c3*	80
RS *c5* + dv + sb	9.4
RS *c5* + dv + sb + np	12.4
RS *c5* + dv + sb + ft + np	15.1
RS dv + sb + *c3*	10.6
RS dv + sb + ft + *c5*	5.8
RS dv + sb + ft + *c3*	3.5
RS dv + sb + ft + *c0*	3.2

TABLE 5.9 – Reconstruction errors for several combinations of hyperparameters for the railway station (RS) dataset. “ft” designates the use of fine-tuning, “sb” the use of subsampling, “dv” the normalisation of the trajectory length, “np” the use of only the velocity in the input vector of the encoder. *c5*, *c3* and *c0* denote respectively the use of the consistency constraint with coefficient $\beta = 10^5$, $\beta = 10^3$, and $\beta = 0$ (i.e, without the consistency constraint). The reconstruction is also given for the STMS dataset as a reference (this method was trained with a consistency parameter of $\beta = 10^5$).

tories. In this section, we present additional experiments on a dataset of biological trajectories. They have been extracted by tracking proteins in a cell in 2D Total Internal Reflection Fluorescence (TIRF) microscopy image sequences [1]. The multiple hypothesis tracking method described in [CBO13] with default parameters was used for trajectory computation. A sample image is given in Figure 5.15. The molecules may follow one of three kinds of motion, namely, confined, Brownian or directed (see Figure 5.16 for an illustration). The confined particles remain in a confined area of the cell, and the particles undergoing Brownian motion evolve erratically in a given area of the cell. On the other hand, particles exhibiting directed motions constantly move in a given direction, leading to longer tracks from the initial location of the particle. The dataset at our disposal contains a small number of directed motions. As a consequence, we consider that directed motions correspond to salient motions in this experiment.

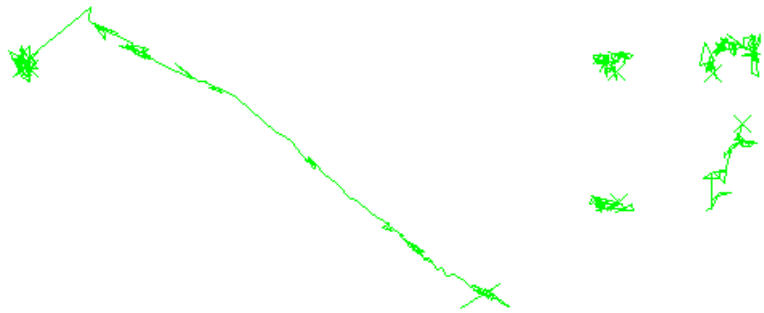


FIGURE 5.16 – Examples of a directed motion (left), two confined motions (middle) and two Brownian motions (right).

To assign trajectories to confined, Brownian and directed classes, we rely on the prediction yielded by the method developed in [Bri+19 ; Bri+20], and we take it as the ground truth. Our objective is to test how our method behaves for this kind of biological trajectories, which are far more irregular than the trajectories of the STMS and even the RS dataset. We divide the data into a training set of 1640 trajectories and a test set of 1169 trajectories. In the test set, 20 trajectories are salient. We pre-train our method on the STMS dataset, and we fine-tune it on the biological trajectories. We consider three levels of the consistency constraint for training, respectively $\beta = 10^5$, $\beta = 10^3$ and $\beta = 0$. The other settings for the fine-tuning step are identical to the ones adopted for the network training on the STMS dataset.

The amount of available data is limited. In order to set λ, we rely on the experiments conducted on the RS dataset. A value of $\lambda = 2$ seemed relevant for most cases (as seen in Table 5.2), and we take it for this experiment. The biological trajectories are erratic for the confined and Brownian motions. It is difficult for the auto-encoder to represent confined or Brownian trajectories without collapsing to almost a point (see Figure 5.17). The variant that correctly find at least

1. Images are provided by UMR 144 CNRS, Institut Curie, PICT IBiSA.

part of the salient trajectories is the one with the intermediate consistency level as summarised in Table 5.10. A too high consistency prevented the learning of relevant codes. No consistency at all does not help. Clearly, the trajectory saliency prediction does not work as well for this biological dataset as for the STMS or the RS datasets.

Perspectives for this type of dataset include searching for a better reconstruction and a more discriminative latent code.

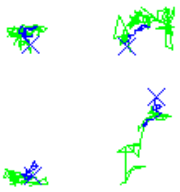


FIGURE 5.17 – Reconstruction of confined (left) and Brownian (right) motions. The ground truth is in green, the reconstruction is in blue.

Method Variant	Precision	Recall	F-measure	FPR	λ
$\beta = 10^3$	0.4	0.5	0.4	0.01	2

TABLE 5.10 – Evaluation of the trajectory saliency detection method on the dataset of biological trajectories.

5.8.6 Code distribution

The consistency constraint was designed to make codes representing similar elements closer. To better understand the effects of this constraint, we computed the empirical distribution of the learned codes with and without the consistency constraint on the STMS validation dataset. The components of each code are plotted as histograms in Figure 5.18. The components corresponding to a training without consistency are denoted f_i, and those with consistency are denoted g_i. Their values lie in $[-1, 1]$.

Without the consistency constraint, all the code components vary largely. Two main patterns stand out. Either the code values are spread in [-1,1], or the code values are restricted to positive or negative ones. When the consistency constraint is applied, the variability is far lower. In almost all cases, the difference between the smallest and largest values is smaller than 0.5, and for several components, the predicted value is practically constant.

The consistency constraint has then a clear impact on the codes. Without it, the network tends to take profit of all the possible values to reconstruct the trajectories with high precision.

When enforcing the constraint, we end up with far smaller variations and only the main significant information is stored. Local variations inherent to each trajectory are more likely to be discarded.

5.9 Conclusion

We have defined a new framework for trajectory saliency detection based on a recurrent auto-encoder. With a careful design of the training loss including a consistency constraint, the framework remains weakly supervised, while still being able to find saliency even when the difference with normality is moderate. By construction, our method is able to handle relative saliency. A trajectory is declared as salient, not because it has never been seen in the training data, but because it departs from its local context. This allows for an easier generalisation to new configurations.

A limitation of the method in its current form is a scenario in which the camera is in motion, or has an inclination that generates perspective in the scene. In this case, the perceived motions will be impacted. The way the consistency is applied may then need to be adapted to remain relevant to the motions that are observed. Similarly, the decision procedure should probably be adapted as well to take into account the modifications of the observed motions.

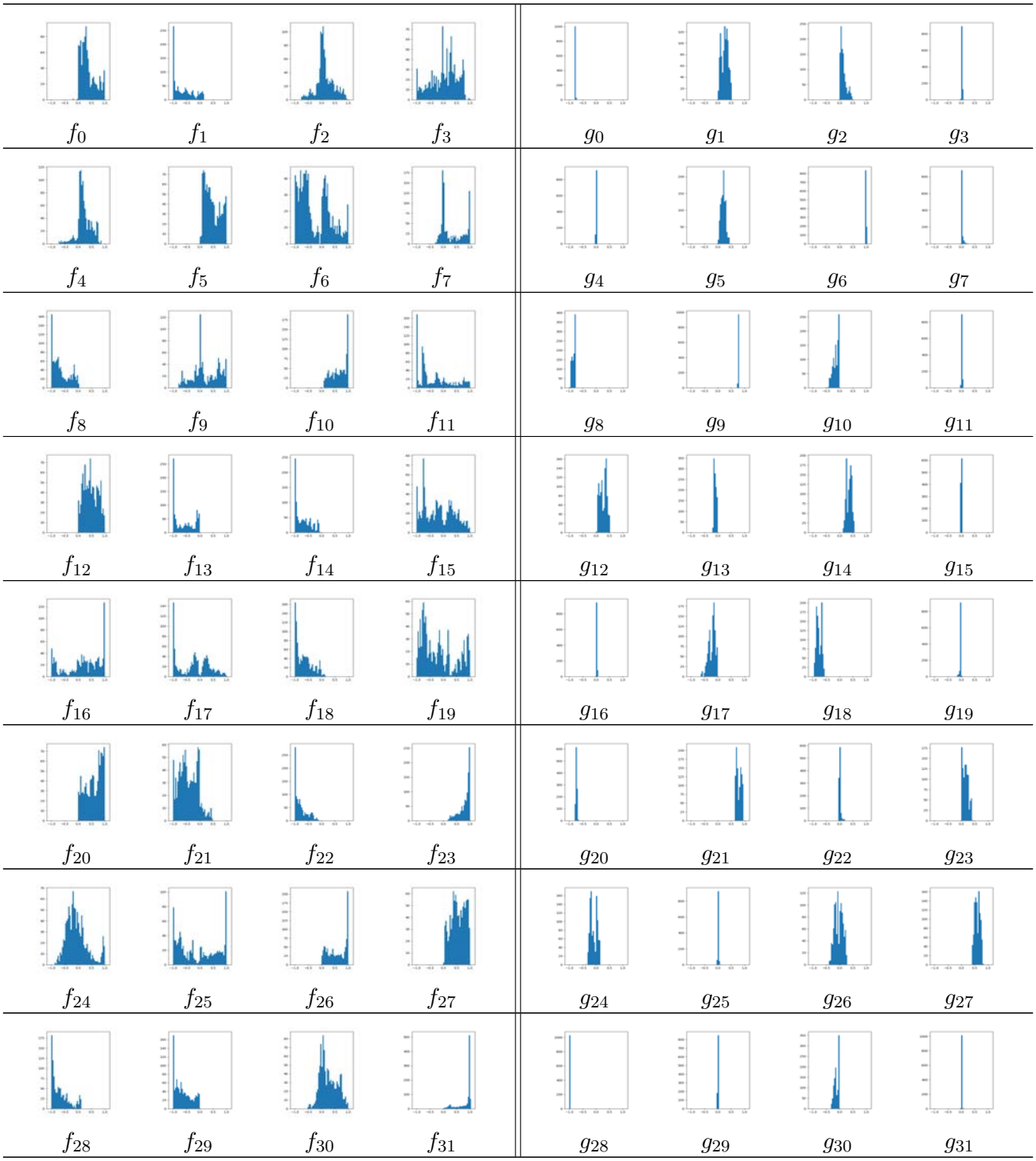


FIGURE 5.18 – Histograms of the computed components of the trajectory codes, after training on STMS dataset. The f_i components on the left correspond to a training without the consistency constraint, and the g_i on the right correspond to a training with the consistency constraint. The dispersion of the components is far more limited with the consistency constraint as expected.

CHAPTER 6

CONCLUSION

6.1 Contributions

In this thesis, we have investigated three complementary aspects of motion saliency, namely, motion saliency detection, motion saliency map estimation, and trajectory saliency estimation. We have designed and evaluated methods to handle these three issues.

6.1.1 Motion saliency detection

First, we have considered the task of temporal motion saliency detection. This problem consists in predicting for each frame of a video if motion saliency is present. It amounts to a pre-attention mechanism to trigger further processing or actions, only when it is necessary. To our knowledge, this has not been investigated so far. Existing methods for saliency map estimation simply assume that saliency is present. To solve this problem of motion saliency detection, we have defined a method based on two main steps. First, the dominant camera motion is compensated. Then, the classification is done with a convolutional neural network. Among the variants tested, the best performing one, denoted RFS-Motion2D, relies on optical flow. The compensation of the dominant motion is done by subtracting the dominant flow to the optical flow, which produces the residual flow used as input to the classification network. Experimental results show that our method is able to properly detect motion saliency, even for scenes in which static object present a strong apparent motion due to their location in the foreground of the scene and camera motion. We reach an accuracy of 87.5% on our real dataset, and even 93.3% on the DAVIS2016 dataset.

6.1.2 Motion saliency map estimation

Second, we have addressed the problem of motion saliency map estimation, which enables to locate motion saliency in each video frame. To this end, we have designed an original method based on optical flow inpainting. The first step of this method consists in extracting salient region candidates. This is achieved by extracting motion boundaries from the computed optical flow, and by processing and refining them. Then, the optical flow in these regions is inpainted from

the surrounding flow. The idea is that a high difference between the reconstructed flow and the flow computed originally indicates the presence of motion saliency. The saliency map is finally computed based on this residual flow. The experiments have shown that our method compares favourably with existing methods.

6.1.3 Trajectory saliency

Third, we have considered the problem of trajectory saliency estimation. Trajectories naturally allow one to handle motion saliency that may appear progressively over long time instants. Our method first estimates an adequate latent representation of trajectories. This representation is obtained with a recurrent auto-encoder network. The loss consists of a combination of two elements that are the classical reconstruction loss of the auto-encoder and a consistency constraint. We have added this second constraint to express that normal trajectories are expected to be similar, and should then be represented with similar codes. The overall setting is almost unsupervised. We just need to check first that most trajectories of the training dataset are normal, that is, follow a similar motion pattern, to apply the consistency constraint. The classification into salient and non-salient trajectories is based on the comparison of the latent codes to the median code in the embedding space. Saliency is characterised by singular codes that are distant from codes representing normal samples. Experiments on synthetic and real datasets show that our method yields good performance. In particular, the consistency constraint allows us to get significant improvements. This is more obvious when the difference between saliency and normality is less pronounced.

6.2 Perspectives

Several ideas and research directions can be explored to improve and extend the work presented in this thesis. A brief discussion around these directions is proposed below.

6.2.1 Possible improvements of the motion saliency detection method

Our method for temporal motion saliency detection includes a step whose role is to compensate the dominant motion due to the camera displacement. To achieve this, the dominant motion is approximated with an affine model. This proved to be sufficient to get good classification results, but it is well known that an affine model is not able to estimate well all the kinds of motions that may be encountered in practice, in particular for complex scenes. One perspective would then be to investigate more sophisticated approaches to estimate the dominant motion. We could for instance adopt a piece-wise estimation of the parametric model, or we could build a learning-based framework to estimate a better representation of the dominant motion.

Our method uses only two video frames. A way to improve the classification would be to introduce temporal regularisation as a way to smooth the estimation over time. More generally, taking into account longer time intervals would allow us to gain more information, and could lead to still improved results.

6.2.2 Distinguishing depth saliency in the motion saliency map estimation

For the estimation of motion saliency maps, our method based on optical flow makes no distinction between "depth saliency" and "true" motion saliency. Depth saliency is due to static objects located in the foreground of the scene and likely to exhibit larger motion than the rest of the static scene when the camera is moving. If the goal is to get true motion saliency exclusively, a first solution is to first apply the temporal motion saliency detection method, and to compute motion saliency maps only when motion saliency is indeed present. However, this is not fully satisfying, as depth saliency and true motion saliency might be jointly present in the same video frame, in which case this solution will fail. Then, an alternative would be to investigate a learning-based mechanism to separate motion and depth saliency, which should *a priori* be supervised.

We could even revisit the overall framework of motion saliency map computation using recent deep learning methods, both for the mask candidate generation and the flow inpainting. One possibility would be to resort to unsupervised adversarial networks as done in [Yan+19] for the detection of independently moving objects in videos. However, this method does not address the depth saliency issue.

6.2.3 Possible improvements of our trajectory saliency estimation framework

Our framework for trajectory saliency detection could be improved, in particular regarding the neural network estimating the trajectory embedding. Modifying the number of layers, their dimension and connectivity may help improve performance.

Another extension would be to handle situations with a moving camera. The way consistency is formulated may need modifications, since the background will also provide trajectories. The decision scheme could be improved too.

6.2.4 Generalisation of our framework for trajectory saliency to other saliency or anomaly detection tasks

Our method for trajectory saliency detection has been designed to extract useful representations of trajectories from low supervision. We have achieved this by combining a consistency

constraint to the classical reconstruction constraint of an auto-encoder. The underlying assumption for the consistency constraint can be valid for other saliency or anomaly estimation tasks, not necessarily linked to motion. Defect detection in textures or anomaly detection of manufactured pieces provide examples of such situations in which most elements in a pack are similar, and in which saliency corresponds to dissimilar elements compared to the majority of normal elements. It would then be straightforward to transpose our framework to such problems, by defining an adequate auto-encoder and by complementing it with a consistency constraint.

6.2.5 Adopting a low-supervision learning paradigm for moving object segmentation

Finally, various problems could be solved by learning and leveraging relevant representations or embeddings. In addition to saliency estimation, other tasks could comprise clustering or classification. Solving these tasks involve ensuring similar elements get a similar representation in the embedding space, and dissimilar elements get a different representation. Instead of considering supervision to learn a relevant representation, which is usually time consuming, it could be more interesting to first analyse data and problems at hand to determine whether a weak supervision can be envisaged. To cite one possible application, a motion that is pronounced enough can be segmented with a relatively good confidence. When an object is moving and well identified, it would be easy to define a loss to constrain the embedding of the pixels of the object to be consistent, and clearly distinct from the embedding of the remaining pixels in the image. It could allow us to extend the method presented in annex to real data. In addition, such an approach would be more easily scalable to handle new classes of objects, by removing the requirement to manually annotate huge amount of data.

APPENDIX A

INVESTIGATION OF RELATIVE MOTION SALIENCY ESTIMATION WITH A 3D VIDEO SALIENCY DATASET

In this appendix, we describe preliminary attempts carried out during the thesis that somehow echo the perspectives outlined in the general conclusion of the manuscript.

In a scene containing mostly static objects, any element with a pronounced motion will be perceived as dynamically salient. Similarly, if the whole scene is perceived as moving with a coherent motion, either due to the camera motion or because the scene is constituted of a very dense flow of elements, any other element with a different motion will be perceived as salient. The problems investigated in Chapters 3 and 4 correspond to these configurations. A more complex variant of motion saliency is relative motion saliency. It consists in identifying in a scene including distinct moving objects the ones whose motion departs from the normal one. The normal motion could be the one shared by the large majority of objects or a reference motion. Salient objects can for instance correspond to objects going in a different direction than the main flow. Chapter 5 precisely considered relative motion saliency, but the method proposed is built for trajectory data and not directly for videos. The relative motion saliency estimation for video has then not been fully addressed.

A prerequisite is the construction of an adequate evaluation setup, with annotated videos displaying relative motion saliency. Additionally, the definition of learning-based methods is possible only if a large enough training dataset is available. As discussed in Chapter 2, deep learning has largely improved performance of computer vision tasks as diverse as object classification, optical flow or generic saliency estimation. This family of methods is likely to be efficient for relative motion saliency estimation as well.

To our knowledge there are no video datasets specifically dedicated to relative motion saliency estimation. A first possibility would be to acquire and manually annotate such a dataset. However, this process would be extremely time consuming, so we did not consider it. A second option consists in generating a dataset of 3D synthetic videos, inspired by works such as [But+12 ; May+16]. The advantage of this approach is that a large amount of data can be

automatically generated and annotated.

The generation of this dataset is described in Section A.1. We then leveraged this dataset to assess the relevance of two methods for relative motion saliency estimation. The first method is based on optical flow and is described in Section A.2. The second method relies on a preliminary segmentation of the scene to extract object trajectories, that are then processed to estimate relative saliency. This method is described in Section A.3.

A.1 Construction of a 3D dataset for relative motion saliency

For the task of optical flow estimation, the synthetic datasets MPI-Sintel [But+12] and FlyingThings3D [May+16] are important assets to evaluate and train methods. They are based on 3D models of the scene and/or of the moving objects. These datasets contributed to the emergence in this field of deep learning-based state-of-the-art methods.

We were inspired by this approach to generate a 3D video dataset for relative motion saliency. To build the MPI-Sintel dataset, the authors relied on an open-source 3D animated movie from which they computed the 2D optical flow. The advantage is that a good realism can be achieved for the video frames, while the ground-truth is exhaustive and accurate. For relative motion saliency estimation, requirements are more constrained. In particular, the video should contain a main flow of objects following the said normal motion, with a few additional objects undergoing salient motion. Consequently, existing open source movies such as Sintel cannot be easily leveraged to generate a dataset for our objective.

Our approach is then closer to the one adopted in [May+16] to generate FlyingThings3D. In this work, the authors used 3D models of objects to which they assigned random motion. It allowed them to fully automatically generate a very large amount of data, the counterpart being a limited realism. Still, motion is well handled for the optical flow estimation task. We similarly exploit 3D models of objects that we randomly draw from the ShapeNet dataset [Cha+15] that contains a total of 63300 models. The large number of models ensures a high diversity of the generated videos. Randomly drawing objects does not ensure semantic relevance of the general scene content. However, it can be viewed as an advantage to guarantee that saliency is only due to motion. To further increase diversity, we may replace textures provided with the objects by procedural textures adapted from [Tor]. We used the open-source software Blender [Ble] to render videos with these models and textures.

In contrast to FlyingThings3D, motions must be “interpretable”, in the sense that human observers must be able to immediately identify salient elements when viewing the video. This implies that objects cannot arbitrarily fly as for FlyingThings3D. Instead, we defined a ground plane on which we materialised a road. The objects are moving on the road, and the normal objects have a similar motion direction and magnitude. The salient objects either go faster

(approximately twice as fast) or go in the opposite direction to the main flow. Salient objects are randomly included. On average, one object out of twenty is salient. In practice, this means that a salient object is not necessarily present in every frame. The video is filmed from a static camera placed on the side of the road or above the road (see Figure A.1 for an illustration). The camera is slightly inclined to generate perspective in the scene. The direction of the main motion and its magnitude randomly vary from one video to the other. We generated 399 training videos and 40 test videos of 240 frames each, for a total of 105360 frames. The resolution of these videos is 960x540. A computing grid with several GPU devices was used to produce these videos.

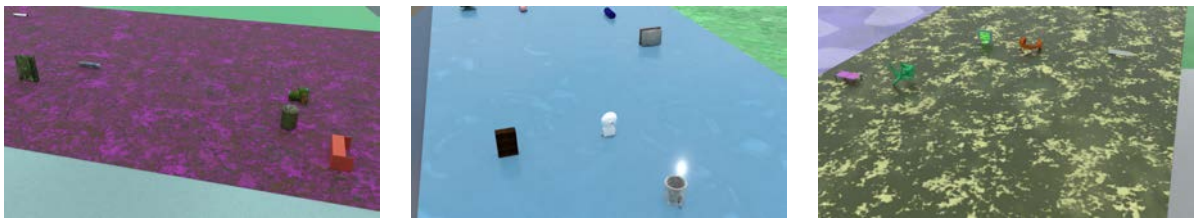


FIGURE A.1 – Samples from the 3D synthetic video dataset for relative motion saliency. Each video depicts a flow of objects, and most of them undergo a similar normal motion. The field of view of the camera allows us to encompass a sufficiently large area, and the apparent size of the objects in the image is limited.

A.2 Approach based on optical flow

The first approach we investigate to estimate relative motion saliency is based on optical flow. We start by describing it in Section A.2.1, before reporting experimental results in Section A.2.2.

A.2.1 Optical flow-based CNN method

Optical flow has proven to be useful for generic saliency estimation, as we have seen in Chapter 2, but also more specifically for motion saliency estimation, as demonstrated in Chapters 3 and 4. Our first approach for relative motion saliency estimation from video sequences is again based on optical flow. Compared to the motion saliency problem addressed in Chapter 4, relative motion saliency with three groups (background, normal independent motion, salient independent motion) is expected to be harder to handle. As a consequence, we prefer to resort to a deep network.

A.2.1.1 Network architecture

Similarly as in Chapters 3 and 4, we are interested in motion saliency only. As a consequence, we do not include an appearance stream in our network. Instead, the prediction will be made from a unique stream that takes as input optical flow. The optical flow itself will of course be estimated from a pair of images. However, optical flow is a dense field representing by nature the full motion information, and it does not reveal any appearance cues.

The exact specification of the network is provided in Figure A.2. The input optical flow is again computed with FlowNet 2.0 [Ilg+17]. Our network aims to produce a prediction of the same resolution as the input image. Then, we adopt design principles from the U-net architecture [RFB15]. More precisely, apart from a few details, our network architecture is almost identical to the one proposed in [TAS17]. In this paper, the authors attempt to segment moving objects from optical flow, which is related to relative motion saliency estimation. To limit the computational burden, the 960x540 images of our dataset are rescaled to a 480x270 resolution to be fed to the network. A final upsampling step is used to reach the same resolution for the output map as for the input image.

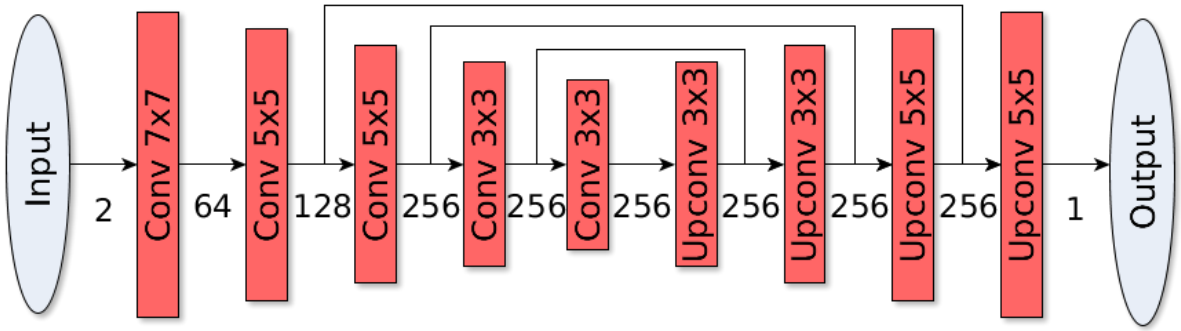


FIGURE A.2 – Network we use for relative motion saliency map estimation. The input is the 2-channel optical flow. The first five convolutional layers are followed by batch normalisation, ReLU and max pooling. The arrows linking the encoding and the decoding part of the network denote the concatenation of the feature maps. The layers denoted by “upconv” are transposed convolutions with a stride of 2. They are followed by batch normalisation and ReLU, except for the last one. A succession of ReLU and hyperbolic tangent is used instead to predict the saliency map.

A.2.1.2 Network training

For training the network, we have defined two loss functions. The first one is simply the mean squared error between the predicted relative motion saliency map and the (binary) ground truth map. In the ground truth, a value of 0 indicates non salient pixels, and a value of 1 indicates

salient pixels. This loss denoted MSE writes as follows :

$$\text{MSE} = \frac{1}{|\Omega|} \sum_{p \in \Omega} (g(p) - g_{GT}(p))^2, \tag{A.1}$$

with p a pixel of the image domain Ω, g the predicted saliency map and g_{GT} the ground truth saliency map. The videos from our dataset are synthetically generated, so the ground truth g_{GT} is available for each frame.

The second loss is inspired by the focal loss proposed in [Lin+17]. Pixels belonging to moving objects are comparatively rare compared to background pixels, and salient objects among moving objects are themselves rare. The MSE loss function puts much more importance on non-salient elements than on salient elements, due to the large imbalance between the two classes. The goal of the focal loss is to ensure that misclassified elements yield an overall larger gradient than elements that are properly classified. The resulting loss denoted $\mathcal{F}_\gamma$ is defined by :

$$\mathcal{F}_\gamma = \frac{1}{|\Omega|} \sum_{p \in \Omega} |g(p) - g_{GT}(p)|^\gamma. \tag{A.2}$$

In practice, we set $\gamma = 5$. In addition, only frames containing salient elements are taken into account for the backpropagation with $\mathcal{F}_\gamma$.

The Adam algorithm [KB14] is used for optimisation with default parameters and with a learning rate of 10^{-3}. Regarding training, data augmentation is used, with random cropping, resizing, color jitter, and random flipping of input images.

A.2.2 Experimental results

When training is conducted with the MSE loss, the network supplies null saliency maps, by predicting saliency values of practically zero for all pixels. The very large unbalance between the salient and normal classes is a convincing explanation for this behaviour.

The loss function $\mathcal{F}_\gamma$ was specifically designed to make influence of locations with incorrect prediction larger. Indeed, the relative saliency values estimated by the network was no more zero everywhere, as illustrated in Figure A.3. However, most moving objects are highlighted in the motion saliency map, instead of only the truly salient objects. On one hand, we can conclude that the contribution of salient elements was reinforced with $\mathcal{F}_\gamma$ compared to MSE. On the other hand, the network remains unable to really separate non-salient motions from salient motions, and then to correctly estimate relative motion saliency maps.

Estimating relative motion saliency from optical flow is then not as easy as it was expected. First of all, moving elements, including salient ones, are small objects in our dataset. A more problematic issue is the nature of the motion saliency here. Speed and direction of normal motion vary from video to video. Moreover, the inclination of the camera generates perspective

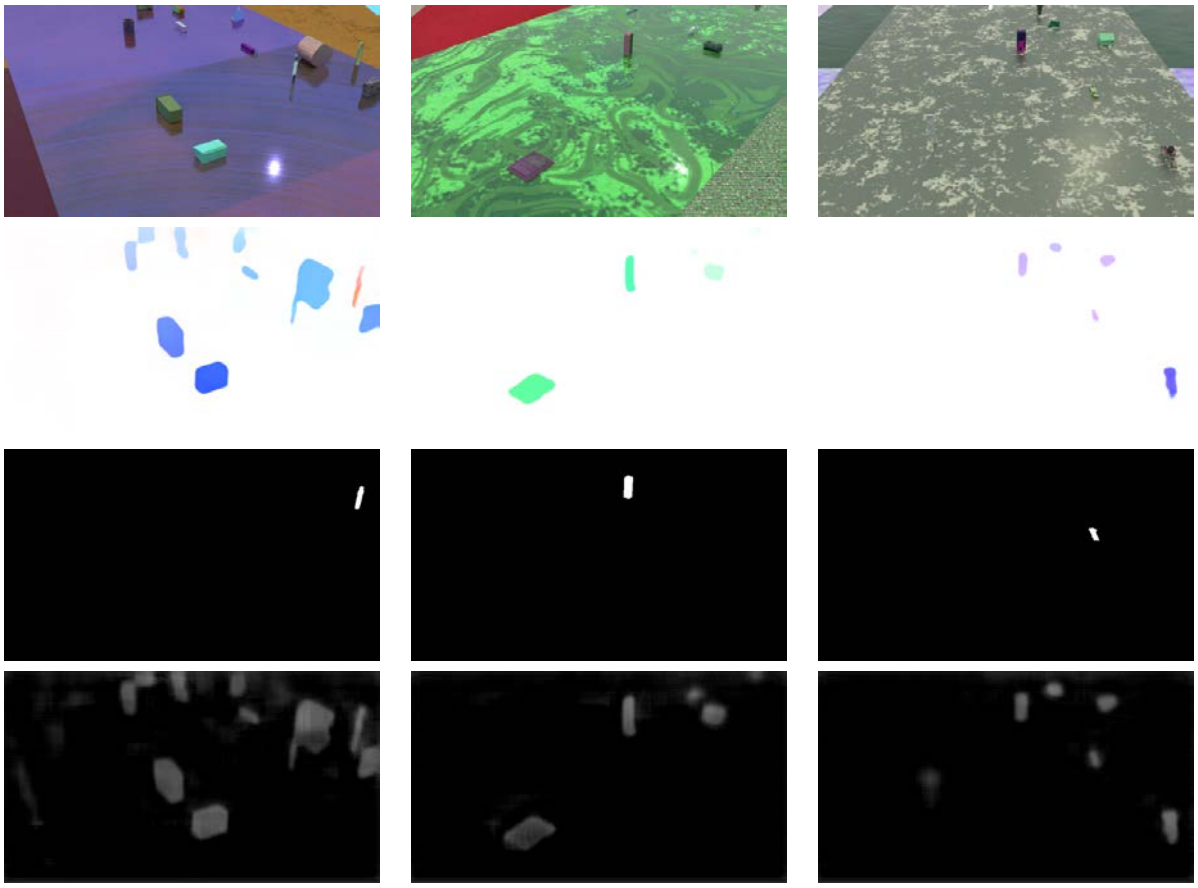


FIGURE A.3 – From top to bottom for each column : input color image, optical flow estimated with FlowNet 2.0, relative motion saliency ground truth, relative motion saliency map estimated with our network based on optical flow. Practically, all moving objects are highlighted in the saliency map, instead of only the salient moving objects.

in the images. Although it poses no problem for human observers to correctly identify the salient moving element, simply processing optical flow with a convolutional neural network is not sufficient. Indeed, the network is expected to compare all motions present in the video and to highlight the singular ones, while being robust to noise present in the estimated optical flow field. This is more complex than predicting the same score for the same pattern, which is typically done for classification. A moving object is salient compared to its surrounding moving objects, not on its own.

Leveraging optical flow did not lead to convincing results for relative motion saliency estimation. This suggested to search for a different approach to solve this problem.

A.3 Approach based on full scene segmentation and trajectories

We will present our second approach for relative motion saliency estimation in Section A.3.1, and report experimental results in Section A.3.2.

A.3.1 Segmentation-based method

The first method was not able to estimate relative motion saliency because it did not succeed in comparing efficiently different motions to find salient ones. The second method will attempt to list all the motions present in the video. To this end, we will first segment the scene into objects. By tracking the position of the objects throughout the video, we will get a list of trajectories used as motion categorization to extract motion saliency.

A.3.1.1 Non-semantic scene segmentation

For relative motion saliency, the notion of object is relevant, since it allows us to group and compare motions. Yet, our objective is to deal with the most generic form of relative motion saliency. Therefore, we do not want to rely on appearance cues to indicate which objects are dynamically salient and which are not. It is common for object trackers to be specialised into a given class of objects to track, for instance pedestrians or cars (see for instance the survey [Cia+20]). This is clearly not compatible with our objective. Instead, the deep metric learning paradigm will inspire us to perform non-semantic scene segmentation and object tracking.

More specifically, our objective will be to estimate with a deep network an embedding $c(p) \in \mathbb{R}^n$ for each pixel p, such that pixels belonging to the same object are represented by a close embedding, and pixels belonging to a different object are represented by a distinct embedding. If we achieve this, the segmentation into different objects should be easy to get with an adequate clustering algorithm. To get a better intuition of why the estimation of an embedding map with these properties should be possible, let us consider the two following points. First,

deep networks can be considered as universal function approximators from their input space to their output space [HSW+89]. The function to be approximated should verify some properties of regularity, but this is generally the case for functions considered in practice. Second, human observers are able to segment natural scenes into objects. Then, it seems reasonable to make the hypothesis that there exists a function mapping natural images to an embedding space, that would enable to separate the different objects. If we take another perspective, we can consider the embedding space to be a feature space with n different features. Even in the case of binary descriptors (value of 0 or 1 for each feature), we can see that n features would allow us to represent up to 2^n different objects. It yields for instance a total of about 3.10^{38} actual ID possibilities for $n = 128$. The embedding space quickly becomes vast enough with increasing values to represent all the objects that can be observed in practice. Above all, it will be easier for the network to assign distinct enough IDs to every object from an embedding lying in $\mathbb{R}^n$ rather than in $\mathbb{R}$.

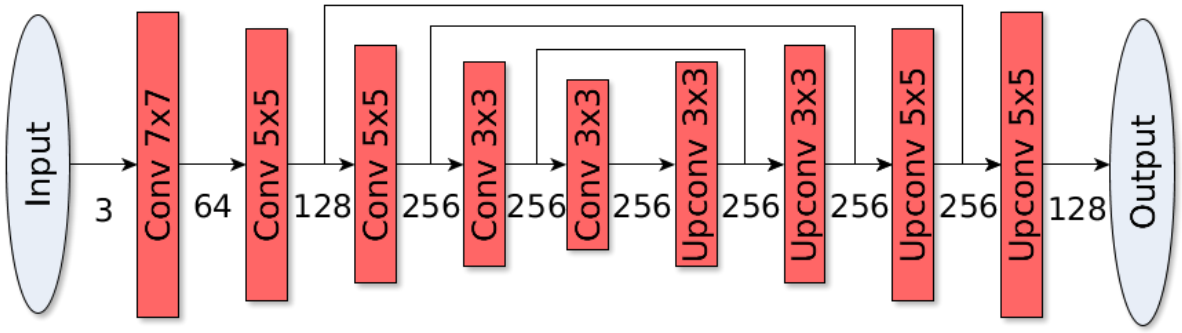


FIGURE A.4 – Network we use for scene segmentation. The input is a RGB image. The first five convolutional layers are followed by batch normalisation, ReLU and max pooling. The arrows linking the encoding and the decoding part of the network denote the concatenation of the feature maps. The layers denoted by "upconv" are transposed convolutions with a stride of 2. They are followed by batch normalisation and ReLU, except for the last one which directly provides the embedding map.

For non-semantic scene segmentation, we leverage a network similar to the one used in Section A.2. This network is displayed in Figure A.4. There are few minor differences between the two frameworks. The input is no longer the 2-channel optical flow but the RGB color image instead. Also, we replace the final 1-dimensional saliency map by a n-dimensional map that will be used to represent objects. In the experiments, we set $n = 128$. We chose this value, as preliminary experiments showed that it could handle a large diversity of objects with a reasonable computational load. In this case, we also want to supply an output map of the same resolution as the input image.

Pixels belonging to the same object should be affected close embeddings, and pixels be-

longing to different objects with distinct embeddings in $\mathbb{R}^n$. For the network training, we define a loss function composed of two terms, respectively to account for consistency and separation. We build batches of unrelated frames for our supervised training procedure. We use index i to denote an object O_i in the set $\mathcal{O}$ of all objects. The term "object" must be taken in a general sense no semantic is involved. The set $\mathcal{O}$ is known since we are dealing with a synthetic training set. During the generation of the videos, the location of all objects is registered, and it allows us to get the precise locations of the objects.

The first term of the loss is defined as follows. First, we define for each object O_i the function :

$$\mathcal{L}_{\mathcal{C}i} = \sum_{p \in O_i} d(\bar{c}_i, c(p))^2, \tag{A.3}$$

where d is the $\mathcal{L}_2$ distance, p a pixel, $c(p)$ the embedding predicted for pixel p and $\bar{c}_i$ the average embedding over O_i :

$$\bar{c}_i = \frac{1}{|O_i|} \sum_{p \in O_i} c(p). \tag{A.4}$$

Initially, the embeddings will probably be unrelated. However, a state of lower energy can be reached during training by making the embeddings of the pixels of O_i closer to a common value. Finally, the full consistency loss writes, with N the number of pixels in the image :

$$\mathcal{L}_{\mathcal{C}} = \frac{1}{N} \sum_{O_i \in \mathcal{O}} \mathcal{L}_{\mathcal{C}i}. \tag{A.5}$$

The second term of the loss whose role is to enforce the separation between the different objects is defined as follows. First, we introduce for each pair $(O_i, O_j) \in \mathcal{O}^2$, with $O_i \neq O_j$:

$$\mathcal{S}_{i,j} = \max(0, \delta - d(\bar{c}_i, \bar{c}_j)), \tag{A.6}$$

with δ the distance value over which two embeddings would not belong to the same object, and $\bar{c}_i$ and $\bar{c}_j$ the respective means over objects O_i and O_j . We set $\delta = 10$ in practice.

The separation loss is then :

$$\mathcal{L}_{\mathcal{S}} = \frac{1}{|\mathcal{O}| - 1} \sum_{(O_i, O_j) \in \mathcal{P}(\mathcal{O})} S_{i,j}, \tag{A.7}$$

with $\mathcal{P}(\mathcal{O})$ all pairs of distinct elements of $\mathcal{O}$. The denominator $|\mathcal{O}| - 1$ is intended to reduce the dependency of $\mathcal{L}_{\mathcal{S}}$ in the number of objects. If we had $n = 1$, we could possibly assume that the embeddings would sequentially lie along the real line, with two neighbours for each object, excepted for the first and last one with respectively the lowest and highest embeddings.

Then, there would be $|\mathcal{O}| - 1$ pairs of objects for which $\mathcal{S}_{i,j}$ is non-zero. For higher values of n, it is not clear how $\mathcal{L}_\mathcal{S}$ will depend on $|\mathcal{O}|$ during training. We then made the choice to keep this normalisation by assuming a linear dependency in $|\mathcal{O}|$.

The final expression of the loss is then :

$$\mathcal{L} = \sum_{\mathcal{B}} \left(\mathcal{L}_\mathcal{C} + \mathcal{L}_\mathcal{S} \right), \tag{A.8}$$

where $\mathcal{B}$ designates a batch of images.

A.3.1.2 Tracking algorithm

From the embedding map with values lying in $\mathbb{R}^n$, we build an object map along time. It amounts to tracking the objects present along time. This is achieved in three main steps.

— The first step performs the clustering of the embeddings $c(p)$ to get the segmented objects. First, the object segmentation is realised for a given frame. We have experimentally observed that the background is represented with embeddings close to zero, and that embeddings of objects lie close to the sphere whose centre is the origin and whose radius is $\delta = 10$, the value used for the separation loss $\mathcal{L}_\mathcal{S}$. Let us mention that the background is included in the object list but no specific treatment is dedicated to it. The representation of the background with an embedding close to zero is only driven by the loss. We then consider that pixels p represented with an embedding $c(p)$ such as $||c(p)||_2 < r$ belong to the background (r is set to 8). All the remaining pixels are processed the following way. The pixel whose embedding c_h has the highest norm, serves as a seed. All pixels p with embeddings $c(p)$ such that :

$$\frac{c_h \cdot c(p)}{||c_h||_2 \cdot ||c(p)||_2} > m, \tag{A.9}$$

with $m = 0.8$ are stated as belonging to the same object. This criterion is preferred to a $\mathcal{L}_2$ distance due to the disposition of the embeddings on a sphere. This process is reiterated until all pixels are classified. In the successive iterations, the remaining unclassified pixel whose embedding has the highest norm is taken as seed. Finally, each cluster, that is, the segmented object in the test set, is represented by $\bar{c_i}$, the average embedding computed in this cluster. The segmented object is denoted by Ω_i.

— Once the segmentation of frame $\mathcal{M}_t$ is achieved, a first re-identification step is conducted. The role of the re-identification is to associate objects between successive instants. Despite trained on single images, the embeddings estimated are quite stable over time. Each object Ω_i is compared with all the objects extracted from $\mathcal{M}_{t-1}$. More precisely, the mean code $\bar{c_i}$ representing Ω_i is compared to all the mean codes $\bar{c_j}$ of the past objects.

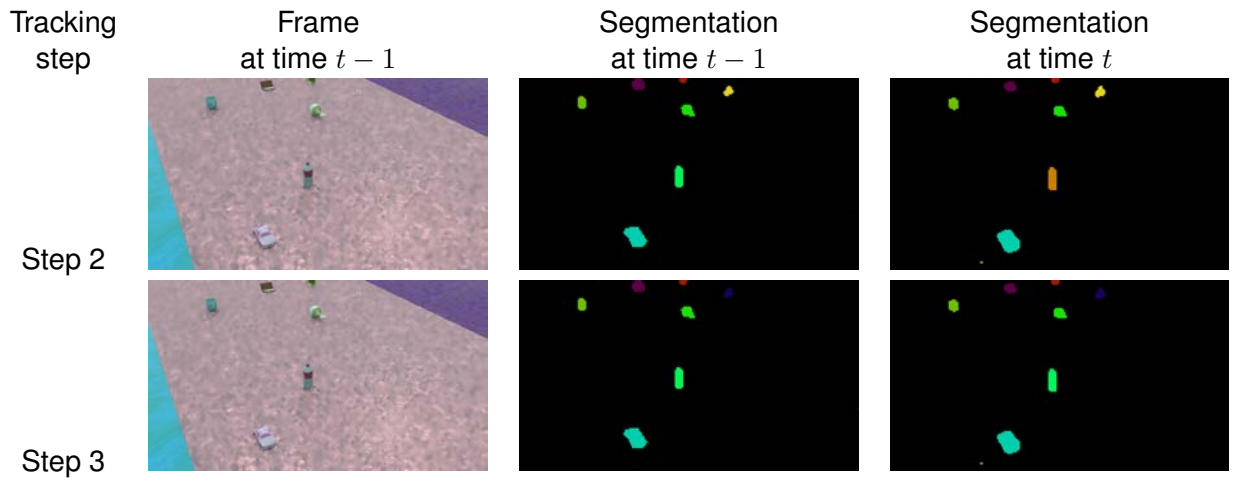


FIGURE A.5 – Illustration of the re-identification algorithm. Each color represents a different object (the background is black). The step 2 that only relies on the embeddings, re-identifies all the objects between $t-1$ and t, except for the middle one in this sample. Additionally, we observe that different objects are represented by different colors (including the green ones), so that no objects are merged. The step 3 that takes into account the spatial location, successfully re-identified the object in the middle of the image.

If there exists a code $\bar{c_j}$ such that :

$$\bar{c_j} \text{ maximises } \frac{\bar{c_i} \cdot \bar{c_j}}{||\bar{c_i}||_2 \cdot ||\bar{c_j}||_2}, \quad \text{provided } \frac{\bar{c_i} \cdot \bar{c_j}}{||\bar{c_i}||_2 \cdot ||\bar{c_j}||_2} > \eta, \tag{A.10}$$

Then, the object Ω_i is re-identified with the past object Ω_j. In practice, we set $\eta = 0.8$. If the criterion A.10 fails, then we consider that Ω_i is a new object. In all cases, the object is still represented by $\bar{c_i}$ in $\mathcal{M}_t$. Retaining the current embedding allows for a smooth evolution of the embedding across time to represent a given object. The resulting segmentation is quite stable as can be seen in Figure A.5, but it can be further improved. Indeed, the clustering step does not guarantee that every object Ω_i is connected. In addition, it may happen that sometimes a given object code changes from one frame to the next one.

— The last step consists in taking into account spatial information to refine the segmentation over time. First, we restart the re-identification for each connected part of an object Ω_i (that may not consist of one single connected part) separately. However, we now add a constraint on the location closeness (in practice within a radius of 30 pixels). Then, a still non-reidentified object Ω_i is identified to the closest object of time $t-1$ (within a radius of 30 pixels). The rationale is that the object speed is limited, so that objects are less likely to go out of the image than to be represented with a slightly shifted embedding. Finally, all objects that were not re-identified are declared new objects.

A.3.1.3 Estimation of relative motion saliency maps

The relative motion saliency estimation is based on the trajectories deduced from the successive scene segmentations. More specifically, each object is represented by the trajectory formed by the successive locations of its centroid. To avoid border effects, the trajectory is interrupted when an object reaches the image border. The limited size and speed of the objects make most of the trajectories be preserved. The segmentation is applied to the whole scene, which includes the static background. Then, a first step consists in removing the static elements. Elements which do not escape a sphere of radius of 60 pixels (in 960x540 images) during the whole video are considered as static. For a better stability, the successive displacements along the trajectories are computed every three frames, that is, between times t and $t+3$ instead of t and $t+1$.

Compared to Chapter 5, the scope of the trajectory processing is here more limited. To estimate saliency, we will build a model of normal motions and compare the trajectories to this model. The displacement is decomposed into magnitude and direction components at each time instant. The direction is given by the unit displacement vector, in order to avoid singularities. We assume that the normal motion over the whole image is regular enough to be approximated by three affine functions, representing respectively the magnitude and the two components of the unit displacement. For a given video, we consider that the three affine maps are stationary over time, and we estimate them with a robust regression method that uses all the computed trajectories as data points. Obviously, the two affine functions accounting for the direction of displacement cannot supply unit vectors everywhere. We simply assume that they can estimate reasonably well the displacement direction everywhere in the image.

Once the three affine maps have been computed, motion saliency is estimated for magnitude and direction separately. A trajectory will be stated as salient for motion magnitude at time instant t, if :

$$\frac{||v(t,p)||}{v_m(p)} > \lambda_m, \tag{A.11}$$

where $v_m(p)$ is the displacement magnitude provided by the corresponding affine function at location p of the trajectory at time t, that is, the centroid of the tracked object at time t, $v(t,p)$ is its displacement vector at time t and λ_m is set to 1.5.

Similarly, a trajectory will be stated as salient for motion direction at time t if :

$$\frac{v(t,p) \cdot v_d(p)}{||v(t,p)|| \cdot ||v_d(p)||} < \lambda_d, \tag{A.12}$$

where $v_d(p)$ is the vector indicating the direction of the motion at point p, supplied by the corresponding estimated affine functions. λ_d is set to 0.5.

A trajectory is finally labelled as salient for motion magnitude if it was stated so for a majority

of instants throughout the video. Similarly, the trajectory is labelled as salient for displacement direction if it was stated as salient at a majority of time instants. The trajectory is finally considered as salient if it is salient for either motion magnitude or direction.

A.3.2 Experimental results

We will now detail the metrics we used to evaluate the method, before presenting the experimental results.

A.3.2.1 Test protocol

Two groups of metrics are used for the objective evaluation of the method. The first group evaluates motion saliency estimation at the frame level. A frame is considered salient if it contains a salient object. More specifically, we measure the rate of correct classification for salient and non salient frames.

The second group of metrics evaluates the accuracy of the estimated location of salient objects in every image of the video. We compute precision, recall and F-measure at the object level. We consider that true positives are salient objects of the ground truth which intersect salient objects of the prediction, and that false negatives are salient objects of the prediction which do not intersect salient objects of the ground truth. These metrics have been preferred to pixel-level metrics since in our dataset, objects are small and saliency is a rare event. Precision, recall and F-measure based on the pixel-level prediction would then be highly penalised by imprecision in location of object boundaries. In contrast, the object-level metrics allow us to know whether salient objects are found in the proper area in the image, while being tolerant to small imprecisions. Let us stress that the shapes of the segmented objects are close to the shapes of the original objects, as can be seen in Figure A.6. It also demonstrates that we escape from trivial solutions. For instance, a trivial solution could be to predict salient moving objects covering more than half of the image. This would mechanically ensure a high proportion of true positives and a low proportion of false negative with the metrics selected, but would obviously be of very limited interest.

A.3.2.2 Results

The quantitative evaluation for the frame-level and object-level metrics is summarised in Table A.1. It was obtained on the test set that contains 1883 frames with salient moving objects and 7677 frames without any saliency. First, we observe that the classification of salient frames produces almost no false positives, while still allowing to find 83% of the salient frames. The object-level precision, recall and F-measure show that the identified salient moving objects are

Metric	*Score*
Classification accuracy of non-salient frames	99.8%
Classification accuracy of salient frames	83.0%
Object-level precision	0.98
Object-level recall	0.81
Object-level F-measure	0.88

TABLE A.1 – Results of relative motion saliency estimation on the test subset of our 3D synthetic video dataset.

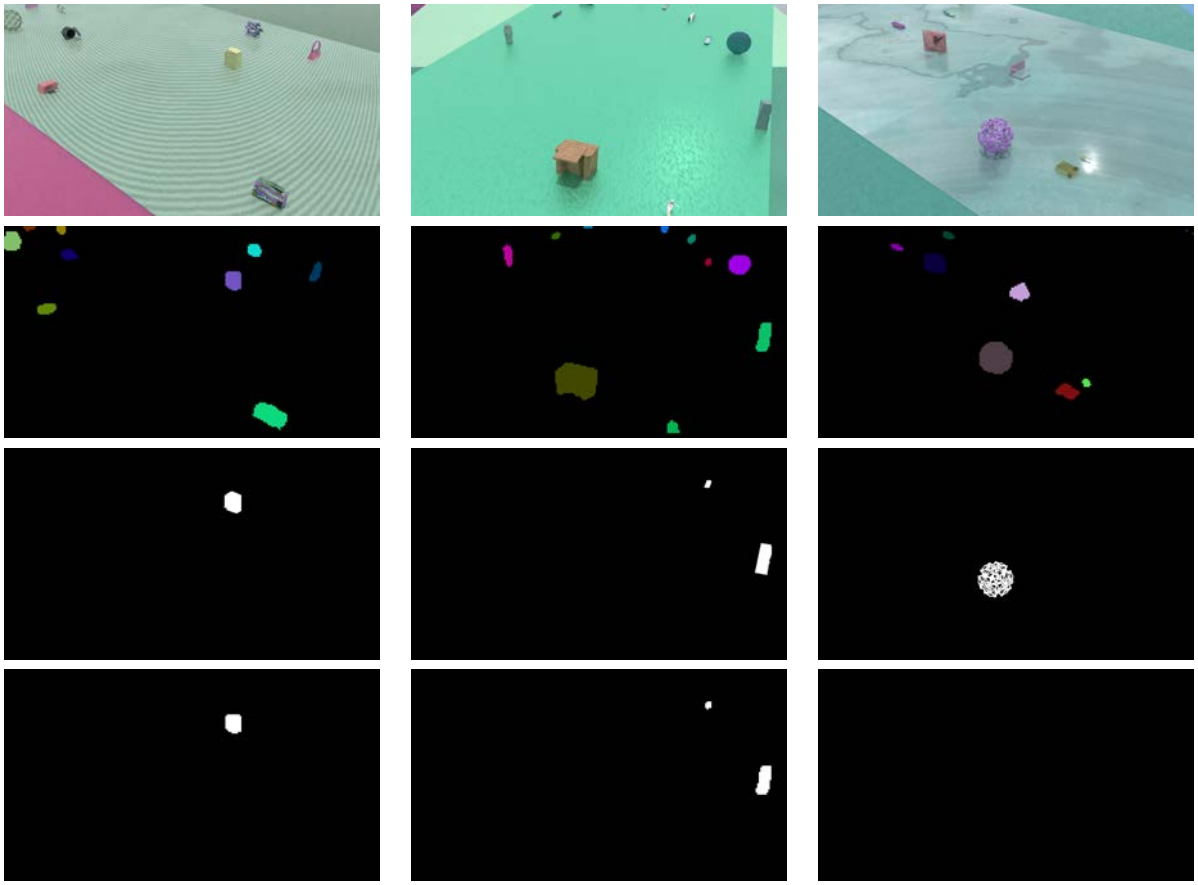


FIGURE A.6 – Samples of qualitative results for relative motion saliency estimation. From top to bottom in each column : video frame, predicted segmentation map (the background is in black), ground-truth saliency map and predicted saliency map. Note that in all cases, the segmentation step manages to find and separate well the objects, even if they occupy small parts of the image. For the first two samples (left and middle), the salient elements are properly identified. The third sample (right) represents a failure case, for which the salient object was not successfully tracked.

generally localised at the correct location in the images. The recall score indicates that 81% of the salient objects are found. A more stable tracking, in particular close to the border of the image, could be a way to improve it. Additionally, Figure A.6 provides a qualitative evaluation, which shows that the location of salient moving objects is correctly delimited. Overall, we observe that this approach yields satisfying results for relative motion saliency estimation for our proposed dataset.

A.4 Conclusion

We have built a 3D synthetic video dataset for relative motion saliency, and we have developed two methods to solve this task. The first approach takes optical flow as input of a deep network. Despite a loss function able to mitigate class imbalance, this approach did not succeed in estimating relative motion saliency. In contrast, the second approach supplied satisfying results on our synthetic dataset. It involves object segmentation of the scene that subsequently allows us to extract object trajectories, and consequently estimate relative motion saliency. The notion of objects that compose the scene was finally beneficial. It facilitates the motion identification and comparison.

These experiments raise several questions. First of all, the scene segmentation network that we trained on our synthetic dataset and that was not fine-tuned on real data, provides so far unusable results when applied to natural videos such as the ones of DAVIS. More specifically, most of the scene tends to be considered as the background, with embeddings that are not easily distinguishable. The gap in appearance between our synthetic videos and real videos is too big for an easy generalisation. Yet, works such as [Che+18c ; Voi+19] showed that semi-supervised object segmentation across time with similar approaches can be achieved on real videos. An open issue is whether such segmentations across time can be applied to the whole scene. Furthermore, the segmentation must extract most normal objects to correctly identify the non-salient motion. It must also extract all the salient objects, that otherwise would not be considered during the saliency estimation process. In particular, it remains unclear whether it will be necessary to dispose of exhaustive scene segmentations ground truth or if alternative methods can be developed. The second issue concerns trajectory saliency estimation. The method used in this appendix is hand-crafted and is not expected to be adapted to more subtle kinds of saliency. It could be relevant to combine this segmentation-based approach with the trajectory saliency detection developed in Chapter 5. As a matter of fact, the latter was conducted after the former. The combination of the two is left for future work.

List of publications

Conference paper

L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Unsupervised Motion Saliency Map Estimation Based On Optical Flow Inpainting », in : *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, p. 4469-4473

Journal paper

L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « CNN-based temporal detection of motion saliency in videos », in : *Pattern Recognition Letters* 128 (2019), p. 298 -305, ISSN : 0167-8655, URL : http://www.sciencedirect.com/science/article/pii/S0167865518307153

National conference papers

L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Détection temporelle de saillance dynamique dans des vidéos par apprentissage profond », in : *Reconnaissance des Formes, Image, Apprentissage et Perception 2018 (RFIAP)*, Marne-la-Vallée, France, 2018

L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Estimation non supervisée de cartes de saillance dynamique dans des vidéos », in : *Reconnaissance des Formes, Image, Apprentissage et Perception 2020 (RFIAP)*, 2020

BIBLIOGRAPHY

[Aba+16] M. ABADI et al., « Tensorflow : A system for large-scale machine learning », in : *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, p. 265-283.

[AHP19] J. AMIRIAN, J.-B. HAYET et J. PETTRÉ, « Social ways : Learning multi-modal distributions of pedestrian trajectories with GANs », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[AMG16] J. T. ANDREWS, E. J. MORTON et L. D. GRIFFIN, « Detecting anomalous data using auto-encoders », in : *International Journal of Machine Learning and Computing* 6.*1* (2016), p. 21.

[AMP10] S. ATEV, G. MILLER et N. P. PAPANIKOLOPOULOS, « Clustering of vehicle trajectories », in : *IEEE transactions on intelligent transportation systems* 11.*3* (2010), p. 647-657.

[ARF14] A. ALAHI, V. RAMANATHAN et L. FEI-FEI, « Socially-Aware Large-Scale Crowd Forecasting », in : *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, p. 2211-2218.

[Ayt+18] Ç. AYTEKIN et al., « Spatiotemporal Saliency Estimation by Spectral Foreground Detection », in : *IEEE Transactions on Multimedia* 20.*1* (2018), p. 82-95.

[Bak+18] C. BAK et al., « Spatio-Temporal Saliency Networks for Dynamic Saliency Prediction », in : *IEEE Transactions on Multimedia* 20.*7* (2018), p. 1688-1698, ISSN : 1520-9210.

[Bar+19] B. BARZ et al., « Detecting Regions of Maximal Divergence for Spatio-Temporal Anomaly Detection », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.*5* (2019), p. 1088-1101.

[BBS01] M. BERTALMIO, A. L. BERTOZZI et G. SAPIRO, « Navier-stokes, fluid dynamics, and image and video inpainting », in : *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, t. 1, 2001.

[Ben12] Y. BENGIO, « Practical recommendations for gradient-based training of deep architectures », in : *Neural networks : Tricks of the trade*, Springer, 2012, p. 437-478.

[Ber+00] M. BERTALMIO et al., « Image inpainting », in : *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, p. 417-424.

[Ber+11] J. BERGSTRA et al., « Theano : Deep learning on gpus with python », in : *NIPS 2011, BigLearning Workshop, Granada, Spain*, t. 3, Citeseer, 2011, p. 1-48.

[BFM19] C. BARATA, M. A. T. FIGUEIREDO et J. S. MARQUES, « Multiple Motion Fields for Multiple Types of Agents », in : *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, p. 1287-1291.

[BHW20] A. BAR-HAIM et L. WOLF, « ScopeFlow : Dynamic Scene Scoping for Optical Flow », in : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, p. 7998-8007.

[BL16] P. BIDEAU et E. G. LEARNED-MILLER, « A Detailed Rubric for Motion Segmentation », in : *CoRR* abs/1610.10033 (2016), arXiv : 1610.10033, URL : http://arxiv.org/abs/1610.10033.

[Ble] BLENDER ONLINE COMMUNITY, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, URL : http://www.blender.org.

[BLM16] P. BIDEAU et E. G. LEARNED-MILLER, « It's Moving ! A Probabilistic Model for Causal Motion Segmentation in Moving Camera Videos », in : *European Conference on Computer Vision (ECCV)*, 2016.

[Bo+19] S. BO et al., « Measurement of anomalous diffusion using recurrent neural networks », in : *Physical Review E* 100.*1* (2019), p. 010102.

[Bor18] A. BORJI, « Saliency Prediction in the Deep Learning Era : An Empirical Investigation », in : *CoRR* abs/1810.03716 (2018), arXiv : 1810.03716, URL : http://arxiv.org/abs/1810.03716.

[Bra00] G. BRADSKI, « The OpenCV Library », in : *Dr. Dobb's Journal of Software Tools* (2000).

[Bri+19] V. BRIANE et al., « A sequential algorithm to detect diffusion switching along intracellular particle trajectories », in : *Bioinformatics* 36.*1* (juin 2019), p. 317-329.

[Bri+20] V. BRIANE et al., « A computational approach for detecting micro-domains and confinement domains in cells : a simulation study », in : *Physical Biology* 17.*2* (2020), p. 025002.

[Bru+20] A. BRUCKERT et al., « Deep Saliency Models : The Quest For The Loss Function », in : *Neurocomputing* (2020), ISSN : 0925-2312.

[BT06] N. BRUCE et J. TSOTSOS, « Saliency based on information maximization », in : *Advances in Neural Information Processing Systems (NIPS)*, 2006, p. 155-162.

[But+12] D. J. BUTLER et al., « A naturalistic open source movie for optical flow evaluation », in : *European Conf. on Computer Vision (ECCV)*, sous la dir. d'A. FITZGIBBON ET AL. (EDS.), Part IV, LNCS 7577, Springer-Verlag, oct. 2012, p. 611-625.

[Cam+17] D. CAMPO et al., « Task-dependent saliency estimation from trajectories of agents in video sequences », in : *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, p. 4252-4256.

[Can86] J. CANNY, « A Computational Approach to Edge Detection », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.*6* (1986), p. 679-698, ISSN : 0162-8828.

[CBO13] N. CHENOUARD, I. BLOCH et J. OLIVO-MARIN, « Multiple Hypothesis Tracking for Cluttered Biological Image Sequences », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.*11* (2013), p. 2736-3750.

[CC19] R. CHALAPATHY et S. CHAWLA, « Deep learning for anomaly detection : A survey », in : *arXiv preprint arXiv :1901.03407* (2019).

[CCB98] A. CRÉTUAL, F. CHAUMETTE et P. BOUTHEMY, « Complex object tracking by visual servoing based on 2D image motion », in : *Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, t. 2, IEEE, 1998, p. 1251-1254.

[Cha+15] A. X. CHANG et al., *ShapeNet : An Information-Rich 3D Model Repository*, rapp. tech. arXiv :1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[Che+16] X. CHEN et al., « Infogan : Interpretable representation learning by information maximizing generative adversarial nets », in : *Advances in Neural Information Processing Systems (NIPS)*, 2016, p. 2172-2180.

[Che+18a] C. CHEN et al., « Bilevel Feature Learning for Video Saliency Detection », in : *IEEE Transactions on Multimedia* 20.*12* (2018), p. 3324-3336.

[Che+18b] L. CHEN et al., « DeepLab : Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.*4* (2018), p. 834-848, ISSN : 0162-8828.

[Che+18c] Y. CHEN et al., « Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 1189-1198.

[Che+20] C. CHEN et al., « Improved Robust Video Saliency Detection Based on Long-Term Spatial-Temporal Information », in : *IEEE Transactions on Image Processing* 29 (2020), p. 1090-1100, ISSN : 1941-0042.

[Cho15] F. CHOLLET, *keras*, `https://github.com/fchollet/keras`, 2015.

[Cho+18] K. H. CHOW et al., « Representation Learning of Pedestrian Trajectories Using Actor-Critic Sequence-to-Sequence Autoencoder », in : *CoRR* abs/1811.08069 (2018), arXiv : `1811.08069`, URL : `http://arxiv.org/abs/1811.08069`.

[Chu+17] Q. CHU et al., « Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism », in : *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 4836-4845.

[Cia+20] G. CIAPARRONE et al., « Deep learning in video multi-object tracking : A survey », in : *Neurocomputing* 381 (2020), p. 61-88.

[CMC18] R. CHALAPATHY, A. K. MENON et S. CHAWLA, « Anomaly detection using one-class neural networks », in : *arXiv preprint arXiv :1802.06360* (2018).

[CPT03] A. CRIMINISI, P. PEREZ et K. TOYAMA, « Object removal by exemplar-based inpainting », in : *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* T. 2, IEEE, 2003, p. II-II.

[CR+18] J. CO-REYES et al., « Self-Consistent Trajectory Autoencoder : Hierarchical Reinforcement Learning with Trajectory Embeddings », in : *Proceedings of the 35th International Conference on Machine Learning*, t. 80, PMLR, 2018, p. 1009-1018.

[Cri+13] T. CRIVELLI et al., « Motion Textures : Modeling, Classification, and Segmentation Using Mixed-State Markov Random Fields », in : *SIAM Journal on Imaging Sciences* 6.*4* (2013), p. 2484-2520.

[DC10] H. M. DEE et A. CAPLIER, « Crowd behaviour analysis using histograms of motion direction », in : *2010 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2010, p. 1545-1548.

[DDM19] A. DAVY, A. DESOLNEUX et J. MOREL, « Detection of Small Anomalies on Moving Background », in : *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, p. 2015-2019.

[DHS11] J. DUCHI, E. HAZAN et Y. SINGER, « Adaptive Subgradient Methods for Online Learning and Stochastic Optimization », in : *Journal of Machine Learning Research* 12.*61* (2011), p. 2121-2159, URL : `http://jmlr.org/papers/v12/duchi11a.html`.

[DJXG17] S. DUTT JAIN, B. XIONG et K. GRAUMAN, « FusionSeg : Learning to Combine Motion and Appearance for Fully Automatic Segmentation of Generic Objects in Videos », in : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[Dos+15] A. DOSOVITSKIY et al., « FlowNet : Learning Optical Flow with Convolutional Networks », in : *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 2758-2766.

[DRT18] N. DEO, A. RANGESH et M. M. TRIVEDI, « How would surround vehicles move ? a unified framework for maneuver classification and motion prediction », in : *IEEE Transactions on Intelligent Vehicles* 3.*2* (2018), p. 129-140.

[DT14] B. DEORI et D. M. THOUNAOJAM, « A survey on moving object tracking in video », in : *International Journal on Information Theory (IJIT)* 3.*3* (2014), p. 31-46.

[EE13] E. ERDEM et A. ERDEM, « Visual saliency estimation by nonlinearly integrating features using region covariances », in : *Journal of vision* 13 (mar. 2013).

[EMK17] T. ERGEN, A. H. MIRZA et S. S. KOZAT, « Unsupervised and semi-supervised anomaly detection with LSTM neural networks », in : *arXiv preprint arXiv :1710.09207* (2017).

[End+16] Y. ENDO et al., « Classifying spatial trajectories using representation learning », in : *International Journal of Data Science and Analytics* 2 (juil. 2016).

[Eve+12] M. EVERINGHAM et al., *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*, 2012, URL : `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`.

[Fan+14] Y. FANG et al., « Video Saliency Incorporating Spatiotemporal Cues and Uncertainty Weighting », in : *IEEE Transactions on Image Processing* 23.*9* (2014), p. 3910-3921, ISSN : 1057-7149.

[Fan+18] C. FAN et al., « Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data », in : *Applied Energy* 211 (2018), p. 1123 -1135, ISSN : 0306-2619.

[Fat+17] A. FATHI et al., « Semantic instance segmentation via deep metric learning », in : *arXiv preprint arXiv :1703.10277* (2017).

[FBK15] D. FORTUN, P. BOUTHEMY et C. KERVRANN, « Optical flow modeling and computation : a survey », in : *Computer Vision and Image Understanding* 134 (2015), p. 1-21.

[FC18] J. FRANKLE et M. CARBIN, « The Lottery Ticket Hypothesis : Training Pruned Neural Networks », in : *CoRR* abs/1803.03635 (2018), arXiv : 1803.03635, URL : http://arxiv.org/abs/1803.03635.

[FGS07] S. FERNÁNDEZ, A. GRAVES et J. SCHMIDHUBER, « An Application of Recurrent Neural Networks to Discriminative Keyword Spotting », in : *Artificial Neural Networks – ICANN 2007*, Springer Berlin Heidelberg, 2007, p. 220-229, ISBN : 978-3-540-74695-9.

[FI14] A. FAKTOR et M. IRANI, « Video Segmentation by Non-Local Consensus voting. », in : *BMVC*, t. 2, 7, 2014, p. 8.

[GGCR17] M. GARCÍA-GARCÍA, A. CAPLIER et M. ROMBAUT, « Driver Head Movements While Using a Smartphone in a Naturalistic Context », in : *6th International Symposium on Naturalistic Driving Research*, t. 8, 2017.

[Ghe+19] E. GHERBI et al., « Modèles génératifs, apprentissage automatique, détection d'anomalie », in : *Conférence sur l'Apprentissage automatique (CAP 2019)*, Toulouse, France, juil. 2019, URL : https://hal.archives-ouvertes.fr/hal-02473865.

[Gid+10] S. GIDEL et al., « Pedestrian detection and tracking in an urban environment using a multilayer laser scanner », in : *IEEE Transactions on Intelligent Transportation Systems* 11.*3* (2010), p. 579-588.

[GLU12] A. GEIGER, P. LENZ et R. URTASUN, « Are we ready for Autonomous Driving ? The KITTI Vision Benchmark Suite », in : *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[Goo+14] I. GOODFELLOW et al., « Generative adversarial nets », in : *Advances in Neural Information Processing Systems (NIPS)*, 2014, p. 2672-2680.

[Goy+12] N. GOYETTE et al., « Changedetection.net : A new change detection benchmark dataset », in : *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, p. 1-8.

[Gra+19] N. GRANIK et al., « Single-particle diffusion characterization by deep learning », in : *Biophysical journal* 117.*2* (2019), p. 185-192.

[GSK18] S. GIDARIS, P. SINGH et N. KOMODAKIS, « Unsupervised Representation Learning by Predicting Image Rotations », in : *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018, URL : https://openreview.net/forum?id=S1v4N2l0-.

[Guo+18] F. GUO et al., « Video Saliency Detection Using Object Proposals », in : *IEEE Transactions on Cybernetics* 48.*11* (2018), p. 3159-3170.

[HA15] E. HOFFER et N. AILON, « Deep Metric Learning Using Triplet Network », in : *Similarity-Based Pattern Recognition - Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015, Proceedings*, t. 9370, Lecture Notes in Computer Science, Springer, 2015, p. 84-92.

[HBL08] A. HERVIEU, P. BOUTHEMY et J. LE CADRE, « A Statistical Video Content Recognition Method Using Invariant Features on Object Trajectories », in : *IEEE Transactions on Circuits and Systems for Video Technology* 18.*11* (2008), p. 1533-1543.

[HBL17] A. HERMANS, L. BEYER et B. LEIBE, « In Defense of the Triplet Loss for Person Re-Identification », in : *CoRR* abs/1703.07737 (2017), arXiv : 1703.07737, URL : http://arxiv.org/abs/1703.07737.

[He+16] K. HE et al., « Deep Residual Learning for Image Recognition », in : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, p. 770-778.

[He+17] K. HE et al., « Mask R-CNN », in : *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[HHS18] Y.-T. HU, J.-B. HUANG et A. G. SCHWING, « Unsupervised video object segmentation using motion saliency-guided spatio-temporal propagation », in : *Proceedings of the European conference on computer vision (ECCV)*, 2018, p. 786-802.

[HLT16] J. HU, J. LU et Y. TAN, « Deep Metric Learning for Visual Tracking », in : *IEEE Transactions on Circuits and Systems for Video Technology* 26.*11* (2016), p. 2056-2068, ISSN : 1558-2205.

[Hou+19] Q HOU et al., « Deeply Supervised Salient Object Detection with Short Connections. », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.*4* (2019), p. 815.

[HR20] J. HUR et S. ROTH, « Optical Flow Estimation in the Deep Learning Age », in : *arXiv preprint arXiv :2004.02853* (2020).

[HS81] B. K. HORN et B. G. SCHUNCK, « Determining optical flow », in : *Techniques and Applications of Image Understanding*, t. 281, International Society for Optics et Photonics, 1981, p. 319-331.

[HS97] S. HOCHREITER et J. SCHMIDHUBER, « Long Short-term Memory », in : *Neural computation* 9 (déc. 1997), p. 1735-80.

[HSW+89] K. HORNIK, M. STINCHCOMBE, H. WHITE et al., « Multilayer feedforward networks are universal approximators. », in : *Neural networks* 2.*5* (1989), p. 359-366.

[HTL18] T. HUI, X. TANG et C. C. LOY, « LiteFlowNet : A Lightweight Convolutional Neural Network for Optical Flow Estimation », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 8981-8989.

[Hua+14] C. R. HUANG et al., « Video Saliency Map Detection by Dominant Camera Motion Removal », in : *IEEE Transactions on Circuits and Systems for Video Technology* 24.*8* (2014), p. 1336-1349, ISSN : 1051-8215.

[Hua+17] G. HUANG et al., « Densely Connected Convolutional Networks », in : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 2261-2269.

[Ilg+17] E. ILG et al., « FlowNet 2.0 : Evolution of Optical Flow Estimation with Deep Networks », in : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 1647-1655.

[Jad+17] M. JADERBERG et al., « Population Based Training of Neural Networks », in : *CoRR* abs/1711.09846 (2017), arXiv : 1711.09846.

[JH13] Y. JIA et M. HAN, « Category-Independent Object-Level Saliency Detection », in : *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013, p. 1761-1768.

[JH96] N. JOHNSON et D. HOGG, « Learning the distribution of object trajectories for event recognition », in : *Image and Vision Computing* 14.*8* (1996), 6th British Machine Vision Conference (BMVC), p. 609 -615, ISSN : 0262-8856.

[Jia+13] B. JIANG et al., « Saliency Detection via Absorbing Markov Chain », in : *IEEE International Conference on Computer Vision (ICCV)*, déc. 2013.

[Jia+14] Y. JIA et al., « Caffe : Convolutional Architecture for Fast Feature Embedding », in : *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, Orlando, Florida, USA, 2014, p. 675-678, ISBN : 978-1-4503-3063-3.

[JKK17] Y. JUN KOH et C.-S. KIM, « Primary Object Segmentation in Videos Based on Region Augmentation and Reduction », in : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[JT20] L. JING et Y. TIAN, « Self-supervised Visual Feature Learning with Deep Neural Networks : A Survey », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[JXW17] L. JIANG, M. XU et Z. WANG, « Predicting video saliency with object-to-motion CNN and two-layer convolutional LSTM », in : *arXiv preprint arXiv :1709.06316* (2017).

[KAB15] M. KEUPER, B. ANDRES et T. BROX, « Motion Trajectory Segmentation via Minimum Cost Multicuts », in : *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 3271-3279.

[Kar+16] A. H. KARIMI et al., « Spatio-temporal saliency detection using abstracted fully-connected graphical models », in : *IEEE International Conference on Image Processing (ICIP)*, 2016, p. 694-698.

[KB14] D. KINGMA et J. BA, « Adam : A Method for Stochastic Optimization », in : *International Conference on Learning Representations (ICLR)*, déc. 2014.

[KB19] M. KAYA et H. Ş. BILGE, « Deep metric learning : a survey », in : *Symmetry* 11.*9* (2019), p. 1066.

[KH+09] A. KRIZHEVSKY, G. HINTON et al., « Learning multiple layers of features from tiny images », in : (2009).

[Kim+15] H. KIM et al., « Spatiotemporal Saliency Detection for Video Sequences Based on Random Walk With Restart », in : *IEEE Transactions on Image Processing* 24.*8* (2015), p. 2552-2564, ISSN : 1057-7149.

[KK14] W. KIM et C. KIM, « Spatiotemporal Saliency Detection Using Textural Contrast and Its Applications », in : *IEEE Transactions on Circuits and Systems for Video Technology* 24.*4* (2014), p. 646-659, ISSN : 1051-8215.

[KSH12] A. KRIZHEVSKY, I. SUTSKEVER et G. E. HINTON, « ImageNet Classification with Deep Convolutional Neural Networks », in : *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, NIPS'12, Lake Tahoe, Nevada, 2012, p. 1097-1105.

[Kum+16] A. KUMAR et al., « Ask me anything : Dynamic memory networks for natural language processing », in : *International conference on machine learning*, 2016, p. 1378-1387.

[Lai+20] Q. LAI et al., « Video Saliency Prediction Using Spatiotemporal Residual Attentive Networks », in : *IEEE Transactions on Image Processing* 29 (2020), p. 1113-1126, ISSN : 1941-0042.

[LeC+98] Y. LECUN et al., « Gradient-based learning applied to document recognition », in : *Proceedings of the IEEE* 86.*11* (1998), p. 2278-2324.

[LF14] R. LAXHAMMAR et G. FALKMAN, « Online Learning and Sequential Anomaly Detection in Trajectories », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.*6* (2014), p. 1158-1173.

[LF18] H. LI et Y. FAN, « Non-rigid image registration using self-supervised fully convolutional networks without training data », in : *Proceedings. IEEE International Symposium on Biomedical Imaging*, t. 2018, avr. 2018, p. 1075-1078.

[LH16] N. LIU et J. HAN, « DHSNet : Deep Hierarchical Saliency Network for Salient Object Detection », in : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, p. 678-686.

[LH18] N. LIU et J. HAN, « A Deep Spatial Contextual Long-Term Recurrent Convolutional Network for Saliency Detection », in : *IEEE Transactions on Image Processing* 27.*7* (2018), p. 3264-3274.

[Li+18] Z. LI et al., « VideoLSTM convolves, attends and flows for action recognition », in : *Computer Vision and Image Understanding* 166 (2018), p. 41 -50, ISSN : 1077-3142.

[Li+18a] G. LI et al., « Flow Guided Recurrent Neural Encoder for Video Salient Object Detection », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 3243-3252.

[Li+18b] S. LI et al., « Instance Embedding Transfer to Unsupervised Video Object Segmentation », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 6526-6535.

[Lin+17] T. LIN et al., « Focal Loss for Dense Object Detection », in : *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 2999-3007.

[Liu+14] Z. LIU et al., « Superpixel-Based Spatiotemporal Saliency Detection », in : *IEEE Transactions on Circuits and Systems for Video Technology* 24.*9* (2014), p. 1522-1540, ISSN : 1051-8215.

[Liu+19] P. LIU et al., « SelFlow : Self-Supervised Learning of Optical Flow », in : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, p. 4566-4575.

[LK+81] B. D. LUCAS, T. KANADE et al., « An iterative image registration technique with an application to stereo vision », in : (1981).

[LLB07] O. LE MEUR, P. LE CALLET et D. BARBA, « Predicting visual fixations on video based on low-level visual features », in : *Vision research* 47.*19* (2007), p. 2483-2498.

[LLU20] G. LIU, F. LIU et A. T. USMAN, « Image Feature Extraction under Microscope Based on Visual Saliency », in : *Acta Microscopica* 29.*4* (2020).

[LM+17] O. LE MEUR et al., « Age-dependent saccadic models for predicting eye movements », in : *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2017, p. 3740-3744.

[LMB13] O. LE MEUR et T. BACCINO, « Methods for comparing scanpaths and saliency maps : strengths and weaknesses », in : *Behavior research methods* 45.*1* (2013), p. 251-266.

[LPC09] J. LUO, C. PAPIN et K. COSTELLO, « Towards Extracting Semantically Meaningful Key Frames From Personal Video Clips : From Humans to Computers », in : *IEEE Transactions on Circuits and Systems for Video Technology* 19.*2* (2009), p. 289-301.

[LS16] T.-N. LE et A. SUGIMOTO, « Contrast Based Hierarchical Spatial-Temporal Saliency for Video », in : *Image and Video Technology*, 2016, p. 734-748, ISBN : 978-3-319-29451-3.

[LS18] T.-N. LE et A. SUGIMOTO, « Video Salient Object Detection Using Spatiotemporal Deep Features », in : *IEEE Transactions on Image Processing* 27.*10* (2018), p. 5002-5015, ISSN : 1057-7149.

[LY16] G. LI et Y. YU, « Visual Saliency Detection Based on Multiscale Deep CNN Features », in : *IEEE Transactions on Image Processing* 25.*11* (2016), p. 5012-5024, ISSN : 1057-7149.

[Mah+18] D. MAHAJAN et al., « Exploring the limits of weakly supervised pretraining », in : *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 181-196.

[Man+11] M. MANCAS et al., « Abnormal motion selection in crowds using bottom-up saliency », in : *2011 18th IEEE International Conference on Image Processing (ICIP)*, 2011, p. 229-232.

[Mat+06] Y. MATSUSHITA et al., « Full-frame video stabilization with motion inpainting », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.*7* (2006), p. 1150-1163, ISSN : 1939-3539.

[May+16] N. MAYER et al., « A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation », in : *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 4040-4048.

[May+18] N. MAYER et al., « What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation ? », in : *International Journal of Computer Vision (IJCV)* 126.*9* (2018), p. 942-960, ISSN : 1573-1405.

[MBL18] L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Détection temporelle de saillance dynamique dans des vidéos par apprentissage profond », in : *Reconnaissance des Formes, Image, Apprentissage et Perception 2018 (RFIAP)*, Marne-la-Vallée, France, 2018.

[MBL19a] L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « CNN-based temporal detection of motion saliency in videos », in : *Pattern Recognition Letters* 128 (2019), p. 298 -305, ISSN : 0167-8655, URL : http : / / www . sciencedirect . com / science / article/pii/S0167865518307153.

[MBL19b] L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Unsupervised Motion Saliency Map Estimation Based On Optical Flow Inpainting », in : *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, p. 4469-4473.

[MBL20] L. MACZYTA, P. BOUTHEMY et O. LE MEUR, « Estimation non supervisée de cartes de saillance dynamique dans des vidéos », in : *Reconnaissance des Formes, Image, Apprentissage et Perception 2020 (RFIAP)*, 2020.

[MCA19] A. MHALLA, T. CHATEAU et N. E. B. AMARA, « Spatio-temporal object detection by deep learning : Video-interlacing to improve multi-object tracking », in : *Image and Vision Computing* 88 (2019), p. 120-131.

[MGS14] D. MAHAPATRA, S. O. GILANI et M. K. SAINI, « Coherency Based Spatio-Temporal Saliency Detection for Video Object Segmentation », in : *IEEE Journal of Selected Topics in Signal Processing* 8.*3* (2014), p. 454-462, ISSN : 1932-4553.

[Mou+15] H. MOUSAVI et al., « Analyzing Tracklets for the Detection of Abnormal Crowd Behavior », in : *2015 IEEE Winter Conference on Applications of Computer Vision*, 2015, p. 148-155.

[MYY18] S. MEI, H. YANG et Z. YIN, « An Unsupervised-Learning-Based Approach for Automated Defect Inspection on Textured Surfaces », in : *IEEE Transactions on Instrumentation and Measurement* 67.*6* (2018), p. 1266-1277.

[Nav+19] M. NAVA et al., « Learning Long-Range Perception Using Self-Supervision From Short-Range Sensors and Odometry », in : *IEEE Robotics and Automation Letters* 4.*2* (2019), p. 1279-1286.

[NF16] M. NOROOZI et P. FAVARO, « Unsupervised learning of visual representations by solving jigsaw puzzles », in : *European Conference on Computer Vision (ECCV)*, 2016, p. 69-84.

[NHD17] A. NEWELL, Z. HUANG et J. DENG, « Associative embedding : End-to-end learning for joint detection and grouping », in : *Advances in Neural Information Processing Systems (NIPS)*, 2017, p. 2277-2287.

[NHLM13] M. NARAYANA, A. HANSON et E. LEARNED-MILLER, « Coherent Motion Segmentation in Moving Camera Videos Using Optical Flow Orientations », in : *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013, p. 1577-1584.

[NSK20] M. T. NGUYEN, P. SIRITANAWAN et K. KOTANI, « Saliency detection in human crowd images of different density levels using attention mechanism », in : *Signal Processing : Image Communication* 88 (2020), p. 115976, ISSN : 0923-5965.

[OB95] J.-M. ODOBEZ et P. BOUTHEMY, « Robust Multiresolution Estimation of Parametric Motion Models », in : *Journal of Visual Communication and Image Representation* 6.*4* (1995), p. 348 -365, ISSN : 1047-3203.

[OB97] J.-M. ODOBEZ et P. BOUTHEMY, « Separation of Moving Regions from Background in an Image Sequence Acquired with a Mobil Camera », in : *Video Data Compression for Multimedia Computing : Statistically Based and Biologically Inspired Techniques*, 1997, p. 283-311, ISBN : 978-1-4615-6239-9.

[Oli+18] M. OLIVER et al., « Motion Inpainting by an Image-Based Geodesic AMLE Method », in : *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, p. 2267-2271.

[OMB14] P. OCHS, J. MALIK et T. BROX, « Segmentation of Moving Objects by Long Term Video Analysis », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.*6* (2014), p. 1187-1200, ISSN : 0162-8828.

[Oze+13] S. OZERÉ et al., « Robust parametric stabilization of moving cells with intensity correction in light microscopy image sequences », in : *2013 IEEE 10th International Symposium on Biomedical Imaging*, 2013, p. 468-471.

[Pap+00] C. PAPIN et al., « Tracking and Characterization of Highly Deformable Cloud Structures », in : *Computer Vision — ECCV 2000*, sous la dir. de D. VERNON, Berlin, Heidelberg, 2000, p. 428-442, ISBN : 978-3-540-45053-5.

[Pap+15] G. PAPANDREOU et al., « Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation », in : *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 1742-1750.

[Par+19] S. PAREKH et al., « Weakly Supervised Representation Learning for Audio-Visual Scene Analysis », in : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2019), p. 416-428.

[Pas+19] A. PASZKE et al., « PyTorch : An Imperative Style, High-Performance Deep Learning Library », in : *Advances in Neural Information Processing Systems 32 (NIPS)*, Curran Associates, Inc., 2019, p. 8024-8035, URL : `http://papers.neurips.`

cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Per+16] F. PERAZZI et al., « A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation », in : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, p. 724-732.

[PF13] A. PAPAZOGLOU et V. FERRARI, « Fast Object Segmentation in Unconstrained Video », in : *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013, p. 1777-1784.

[PLN02] D. PARKHURST, K. LAW et E. NIEBUR, « Modeling the role of salience in the allocation of overt visual attention », in : *Vision Research* 42.*1* (2002), p. 107 -123, ISSN : 0042-6989.

[PR+16] J.-M. PÉREZ-RÚA et al., « Discovering motion hierarchies via tree-structured coding of trajectories », in : *Proceedings of the British Machine Vision Conference (BMVC)*, 2016, p. 106.1-106.12.

[PRBB17] J.-M. PÉREZ-RÚA, A. BASSET et P. BOUTHEMY, « Detection and Localization of Anomalous Motion in Video Sequences from Local Histograms of Labeled Affine Flows », in : *Frontiers in ICT, Computer Image Analysis*, Computer Image Analysis (2017).

[QGH18] W. QIU, X. GAO et B. HAN, « Eye fixation assisted video saliency detection via total variation-based pairwise interaction », in : *IEEE Transactions on Image Processing* 27.*10* (2018), p. 4724-4739.

[Raa+20] L. RAAD et al., « On Anisotropic Optical Flow Inpainting Algorithms », in : *Image Processing On Line* 10 (2020), p. 78-104.

[RAS17] I. ROCCO, R. ARANDJELOVIC et J. SIVIC, « Convolutional Neural Network Architecture for Geometric Matching », in : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 39-48.

[Rav+17] M. RAVANBAKHSH et al., « Abnormal event detection in videos using generative adversarial nets », in : *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, p. 1577-1581.

[RB18] P. R. ROY et G. BILODEAU, « Road User Abnormal Trajectory Detection Using a Deep Autoencoder », in : *Advances in Visual Computing*, Springer International Publishing, 2018, p. 748-757, ISBN : 978-3-030-03801-4.

[RB19] P. R. ROY et G. BILODEAU, « Adversarially Learned Abnormal Trajectory Classifier », in : *2019 16th Conference on Computer and Robot Vision (CRV)*, 2019, p. 65-72.

[RFB15] O. RONNEBERGER, P. FISCHER et T. BROX, « U-net : Convolutional networks for biomedical image segmentation », in : *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, p. 234-241.

[RKB04] C. ROTHER, V. KOLMOGOROV et A. BLAKE, « GrabCut -Interactive Foreground Extraction using Iterated Graph Cuts », in : *ACM Transactions on Graphics (SIGGRAPH)* (2004), p. 309-314.

[RLL18] M. RIBEIRO, A. E. LAZZARETTI et H. S. LOPES, « A study of deep convolutional auto-encoders for anomaly detection in videos », in : *Pattern Recognition Letters* 105 (2018), Machine Learning and Applications in Artificial Intelligence, p. 13 -22, ISSN : 0167-8655.

[Rol+17] D. ROLNICK et al., « Deep Learning is Robust to Massive Label Noise », in : *CoRR* abs/1705.10694 (2017), arXiv : 1705.10694.

[Ros58] F. ROSENBLATT, « The perceptron : a probabilistic model for information storage and organization in the brain. », in : *Psychological review* 65.*6* (1958), p. 386.

[Rou+17] P. ROUDOT et al., « Piecewise-Stationary Motion Modeling and Iterative Smoothing to Track Heterogeneous Particle Motions in Dense Environments », in : *IEEE Transactions on Image Processing* 26.*11* (2017), p. 5395-5410.

[RT18] E. RISTANI et C. TOMASI, « Features for Multi-target Multi-camera Tracking and Re-identification », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 6036-6046.

[Ruf+18] L. RUFF et al., « Deep one-class classification », in : *International conference on machine learning*, 2018, p. 4393-4402.

[Rus+15] O. RUSSAKOVSKY et al., « ImageNet Large Scale Visual Recognition Challenge », in : *International Journal of Computer Vision (IJCV)* 115.*3* (2015), p. 211-252.

[RW20] T. RATEKE et A. von WANGENHEIM, « Road obstacles positional and dynamic features extraction combining object detection, stereo disparity maps and optical flow data », in : *arXiv preprint arXiv :2006.14011* (2020).

[Sal+16] T. SALIMANS et al., « Improved techniques for training gans », in : *Advances in Neural Information Processing Systems (NIPS)*, 2016, p. 2234-2242.

[SCB12] W. SABBAR, A. CHERGUI et A. BEKKHOUCHA, « Video summarization using shot segmentation and local motion estimation », in : *Second International Conference on the Innovative Computing Technology (INTECH 2012)*, 2012, p. 190-193.

[Sch+01] B. SCHÖLKOPF et al., « Estimating the support of a high-dimensional distribution », in : *Neural computation* 13.*7* (2001), p. 1443-1471.

[SDC14] M. STROBEL, J. DIEBOLD et D. CREMERS, « Flow and Color Inpainting for Video Completion », in : *Pattern Recognition*, 2014, p. 293-304, ISBN : 978-3-319-11752-2.

[SHT20] M. SHOKRI, A. HARATI et K. TABA, « Salient object detection in video using deep non-local neural networks », in : *Journal of Visual Communication and Image Representation* 68 (2020), p. 102769, ISSN : 1047-3203.

[SKP15] F. SCHROFF, D. KALENICHENKO et J. PHILBIN, « FaceNet : A Unified Embedding for Face Recognition and Clustering », in : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[SKS16] S. SHARMA, R. KIROS et R. SALAKHUTDINOV, « Action Recognition using Visual Attention », in : *2016 International Conference on Learning Representations (ICLR)*, 2016.

[SLW14] J. SHAO, C. C. LOY et X. WANG, « Scene-Independent Group Profiling in Crowd », in : *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, p. 2227-2234.

[Son+18] H. SONG et al., « Pyramid Dilated Deeper ConvLSTM for Video Salient Object Detection », in : *The European Conference on Computer Vision (ECCV)*, 2018.

[SRB12] A. A. SODEMANN, M. P. ROSS et B. J. BORGHETTI, « A Review of Anomaly Detection in Automated Surveillance », in : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.*6* (2012), p. 1257-1272.

[Su+16] H. SU et al., « Crowd scene understanding with coherent recurrent neural networks », in : *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, p. 3469-3476.

[Sun+17] C. SUN et al., « Revisiting Unreasonable Effectiveness of Data in Deep Learning Era », in : *IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 843-852.

[Sun+18] D. SUN et al., « PWC-Net : CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume », in : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Sun+19] C. SUN et al., « VideoBERT : A Joint Model for Video and Language Representation Learning », in : *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, p. 7463-7472.

[Sun+20] D. SUN et al., « Models Matter, So Does Training : An Empirical Study of CNNs for Optical Flow Estimation », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.*6* (2020), p. 1408-1423.

[SZ14] K. SIMONYAN et A. ZISSERMAN, « Two-Stream Convolutional Networks for Action Recognition in Videos », in : *Advances in Neural Information Processing Systems (NIPS)*, 2014, p. 568-576.

[SZ15] K. SIMONYAN et A. ZISSERMAN, « Very Deep Convolutional Networks for Large-Scale Image Recognition », in : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015, URL : http://arxiv.org/abs/1409.1556.

[Sze+15] C. SZEGEDY et al., « Going deeper with convolutions », in : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[TAS17] P. TOKMAKOV, K. ALAHARI et C. SCHMID, « Learning Motion Patterns in Videos », in : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 531-539.

[TBD18] M. TESFALDET, M. A. BRUBAKER et K. G. DERPANIS, « Two-Stream Convolutional Networks for Dynamic Texture Synthesis », in : *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[Tel04] A. TELEA, « An Image Inpainting Technique Based on the Fast Marching Method », in : *Journal of Graphics Tools* 9 (jan. 2004), p. 23-34.

[TH12] T. TIELEMAN et G. HINTON, « Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude », in : *COURSERA : Neural networks for machine learning* 4.*2* (2012), p. 26-31.

[TK91] C. TOMASI et T. KANADE, « Detection and tracking of point features », in : (1991).

[Tor] J. TORNHILL, http://www.blenderinsight.com.

[Voi+19] P. VOIGTLAENDER et al., « FEELVOS : Fast End-To-End Embedding Learning for Video Object Segmentation », in : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, p. 9473-9482.

[VR79] C VAN RIJSBERGEN, « (1979) », in : *Information Retrieval. 2nd edition. London, England : Butterworths* (1979).

[Wan+14] J. WANG et al., « Learning fine-grained image similarity with deep ranking », in : *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, sept. 2014, p. 1386-1393.

[Wan+17] J. WANG et al., « Deep Metric Learning with Angular Loss », in : *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, p. 2612-2620.

[Wan+17] J. WANG et al., « Salient Object Detection : A Discriminative Regional Feature Integration Approach », in : *International Journal of Computer Vision (IJCV)* 123.*2* (2017), p. 251-268, ISSN : 1573-1405.

[Wan+19] Y. WANG et al., « Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization », in : *arXiv preprint arXiv :1906.00917* (2019).

[Wan+19a] N. WANG et al., « Unsupervised Deep Tracking », in : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, p. 1308-1317.

[Wan+19b] W. WANG et al., « Semi-Supervised Video Object Segmentation with Super-Trajectories », in : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.*4* (2019), p. 985-998.

[Wan+19c] X. WANG et al., « Ranked List Loss for Deep Metric Learning », in : *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, p. 5202-5211.

[Wei+19] W. WEI et al., « Saliency prediction via multi-level features and deep supervision for children with autism spectrum disorder », in : *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, IEEE, 2019, p. 621-624.

[WQT15] L. WANG, Y. QIAO et X. TANG, « Action recognition with trajectory-pooled deep-convolutional descriptors », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, p. 4305-4314.

[WSP15] W. WANG, J. SHEN et F. PORIKLI, « Saliency-aware geodesic video object segmentation », in : *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, p. 3395-3402.

[WSS15] W. WANG, J. SHEN et L. SHAO, « Consistent Video Saliency Using Local Gradient Flow Optimization and Global Refinement », in : *IEEE Transactions on Image Processing* 24.*11* (2015), p. 4185-4196, ISSN : 1057-7149.

[WSS18] W. WANG, J. SHEN et L. SHAO, « Video Salient Object Detection via Fully Convolutional Networks », in : *IEEE Transactions on Image Processing* 27.*1* (2018), p. 38-49, ISSN : 1057-7149.

[Wu+20] Y. WU et al., « Mirrored Autoencoders with Simplex Interpolation for Unsupervised Anomaly Detection », in : *arXiv preprint arXiv :2003.10713* (2020).

[Xia+18] H. XIAO et al., « MoNet : Deep Motion Exploitation for Video Object Segmentation », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 1140-1148.

[XRK17] J. XU, R. RANFTL et V. KOLTUN, « Accurate Optical Flow via Direct Cost Volume Processing », in : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, p. 5807-5815.

[Xu+17] D. XU et al., « Detecting anomalous events in videos by learning deep representations of appearance and motion », in : *Computer Vision and Image Understanding* 156 (2017), p. 117 -127, ISSN : 1077-3142.

[Yan+19] P. YAN et al., « Semi-Supervised Video Salient Object Detection Using Pseudo-Labels », in : *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, p. 7283-7292.

[Yan+19] Y. YANG et al., « Unsupervised moving object detection via contextual information separation », in : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, p. 879-888.

[Yao+15] L. YAO et al., « Describing Videos by Exploiting Temporal Structure », in : *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 4507-4515.

[Yao+17] D. YAO et al., « Trajectory clustering via deep representation learning », in : *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, p. 3880-3887.

[Yao+19] D. YAO et al., « Computing Trajectory Similarity in Linear Time : A Generic Seed-Guided Neural Metric Learning Approach », in : *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, p. 1358-1369.

[YHD16] J. J. YU, A. W. HARLEY et K. G. DERPANIS, « Back to Basics : Unsupervised Learning of Optical Flow via Brightness Constancy and Motion Smoothness », in : *Computer Vision – ECCV 2016 Workshops*, 2016, p. 3-10, ISBN : 978-3-319-49409-8.

[YJF98] B.-K. YI, H. V. JAGADISH et C. FALOUTSOS, « Efficient retrieval of similar time sequences under time warping », in : *Proceedings 14th International Conference on Data Engineering*, IEEE, 1998, p. 201-208.

[ZC18] K. ZHANG et Z. CHEN, « Video saliency prediction based on spatial-temporal two-stream network », in : *IEEE Transactions on Circuits and Systems for Video Technology* 29.*12* (2018), p. 3544-3557.

[ZDK19] C. ZHAO, R. DING et H. L. KEY, « End-To-End Visual Place Recognition Based on Deep Metric Learning and Self-Adaptively Enhanced Similarity Metric », in : *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, p. 275-279.

[Zen+18] Y. ZENG et al., « Learning to Promote Saliency Detectors », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 1644-1653.

[Zha+18] X. ZHANG et al., « Progressive Attention Guided Recurrent Network for Salient Object Detection », in : *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, p. 714-722.

[Zhu+18] J. ZHU et al., « Online multi-object tracking with dual matching attention networks », in : *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 366-382.

[Zhu+19] T. ZHUO et al., « Unsupervised online video object segmentation with motion property understanding », in : *IEEE Transactions on Image Processing* 29 (2019), p. 237-249.

[ZN17] Y. ZHU et S. D. NEWSAM, « DenseNet for Dense Flow », in : *CoRR* abs/1707.06316 (2017), arXiv : 1707.06316, URL : http://arxiv.org/abs/1707.06316.

[ZTG12] H.-G. ZIMMERMANN, C. TIETZ et R. GROTHMANN, « Forecasting with Recurrent Neural Networks : 12 Tricks », in : *Neural Networks : Tricks of the Trade : Second Edition*, 2012, p. 687-707, ISBN : 978-3-642-35289-8.

[ZTW12] B. ZHOU, X. TANG et X. WANG, « Coherent Filtering : Detecting Coherent Motions from Crowd Clutters », in : *Computer Vision – ECCV 2012*, 2012, p. 857-871, ISBN : 978-3-642-33709-3.

UNIVERSITÉ DE RENNES 1

Titre : Saillance visuelle dynamique dans des séquences d'images

Mot clés : Saillance du mouvement, Analyse de vidéos, Réseaux de neurones profonds, *Inpainting* du flot optique, Analyse de trajectoires

Resumé : Les travaux de la thèse portent sur l'estimation de la saillance du mouvement dans des séquences d'images. Dans une première partie, nous avons traité un sujet très peu abordé : la détection des images présentant un mouvement saillant. Pour cela, nous nous appuyons sur un réseau de neurones convolutif et sur la compensation du mouvement de la caméra. Dans une seconde partie, nous avons conçu une méthode originale d'estimation de cartes de saillance du mouvement. Cette méthode ne requiert pas d'apprentissage. L'indice de saillance est obtenu par une étape d'*inpainting* du flot optique, suivie d'une comparaison avec le flot initial. Dans un troisième temps, nous nous sommes intéressés à l'estimation de la saillance de trajectoires pour appréhender une saillance progressive. Nous construisons une méthode faiblement supervisée s'appuyant sur un réseau auto-encodeur récurrent, qui représente chaque trajectoire avec un code latent. Toutes ces méthodes ont été validées sur des données de vidéo réelles.

Title : Dynamic visual saliency in image sequences

Keywords : Motion saliency, Video analysis, Deep neural networks, Optical flow inpainting, Trajectory analysis

Abstract : Our thesis research is concerned with the estimation of motion saliency in image sequences. First, we have defined an original method to detect frames in which a salient motion is present. For this, we propose a framework relying on a deep neural network, and on the compensation of the dominant camera motion. Second, we have designed a method for estimating motion saliency maps. This method requires no learning. The motion saliency cue is obtained by an optical flow inpainting step, followed by a comparison with the initial flow. Third, we consider the problem of trajectory saliency estimation to handle progressive saliency over time. We have built a weakly supervised framework based on a recurrent auto-encoder that represents trajectories with latent codes. Performance of the three methods was experimentally assessed on real video datasets.