Unsupervised learning for motion segmentation and motion saliency in videos

Etienne Meunier

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *AST (Signal, Image, Vision)*

Par
Etienne MEUNIER

Unsupervised learning for motion segmentation and motion saliency in videos

Thèse présentée et soutenue à Rennes, le 04 décembre 2023
Unité de recherche : Centre Inria de l'Université de Rennes

Rapporteurs avant soutenance :

Nicolas THOME Professeur, Sorbonne Université
Patrick PEREZ Directeur scientifique HDR, valeo.ai

Composition du Jury :

Président :	Elisa FROMONT	Professeure, Université de Rennes
Examinateurs :	Daniel CREMERS	Professeur, Technical University of Munich
	Ronan FABLET	Professeur, IMT Atlantique
Dir. de thèse :	Patrick BOUTHEMY	Directeur de Recherche, Centre Inria Rennes

Invité(s) :

Renaud FRAISSE Senior Expert, Airbus Defence&Space

Acknowledgements

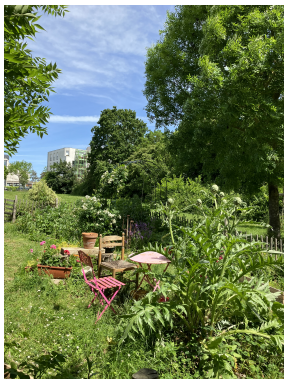
Dans un premier temps, je tiens à remercier mon directeur de thèse, Patrick Bouthemy, pour son accompagnement tout au long de ce travail. C'est grâce à sa bienveillance et à nos échanges que j'ai pu entreprendre les travaux présentés dans ce manuscrit.

Je tiens également à exprimer ma gratitude envers mes rapporteurs, Nicolas Thome et Patrick Perez, pour leur relecture attentive du manuscrit et leurs retours pertinents qui m'ont permis d'améliorer ce travail. Mes remerciements vont également à Elisa Fromont, Daniel Cremers, Ronan Fablet et Renaud Fraisse pour leur participation à mon jury de thèse et leurs retours sur mon travail.

Un grand merci aux membres de l'équipe Sairpico et Odyssey à l'Inria Rennes avec qui j'ai pu avoir des échanges enrichissants, qui m'ont aidé à surmonter certains de mes problèmes et à développer un intérêt pour de nouveaux sujets fascinants.

Je souhaite exprimer ma reconnaissance envers ma famille et mes amis qui m'ont encouragé, soutenu et avec qui j'ai partagé des moments qui me sont chers. Ils ont, par leur présence, rendu possible la rédaction de cette thèse.

Enfin, je tiens à mentionner le potager collectif situé au croisement de la rue des Plantes et de la rue Mirabeau ainsi que les volontaires qui en prennent soin. Tout au long de mon travail de thèse, c'est ici que j'ai pu trouver le calme nécessaire pour écrire et porter un regard nouveau sur mon travail.



J'ai eu la chance extraordinaire de réaliser ma thèse dans ce cadre si encourageant, et je suis fier des trois années que j'ai consacrées à l'étude du mouvement.

Table of Contents

Résumé en Français

Contexte et motivation

Contexte de l'analyse du mouvement

La façon dont nous percevons et interprétons le mouvement nous fournit des informations essentielles sur notre environnement. D'un point de vue fonctionnel, parce qu'un objet en mouvement peut représenter quelque chose de particulièrement digne d'attention, le mouvement nous aide à nous concentrer sur ce qui est important autour de nous. Le mouvement intègre également le fait que le monde qui nous entoure est en train de changer et que nous devons prêter attention à cette version actualisée de notre environnement. Tout comme l'apparence, le mouvement des objets ou des entités qui nous entourent peut faire l'objet d'une interprétation et nous fournir des indications utiles sur la manière d'interagir avec eux.

Dans le contexte de la recherche visant à développer des systèmes de vision par ordinateur et d'intelligence artificielle pour assister les humains, il est crucial de construire des mécanismes capables de percevoir et de représenter le mouvement environnant de manière significative. Toutefois, cette tâche est difficile à reproduire car le système visuel humain est étonnamment doué pour comprendre les mouvements. Par exemple, lorsque nous marchons dans un endroit bondé où se produisent de nombreux mouvements différents, nous sommes capables de remarquer rapidement un seul d'entre eux et de nous concentrer dessus. De même, lorsque nous courons dans une forêt, nous sommes capables d'ignorer la parallaxe, due à notre propre déplacement, de mouvement apparent des arbres qui nous entourent, afin de nous concentrer sur la présence potentielle d'objets se déplaçant indépendamment les uns des autres. Un examen de cette question et quelques explications sur la manière dont le cerveau humain gère cette tâche sont donnés dans des travaux antérieurs [1].

Motivations pour l'analyse du mouvement

L'analyse du mouvement est l'une des principales tâches de la vision par ordinateur et a de nombreuses applications dans une grande variété de domaines. Tout d'abord, la segmentation d'objets en mouvement a des applications dans le montage vidéo, où nous pouvons sélectionner des objets en fonction de leur mouvement, mais aussi en robotique ou en conduite autonome, où le mouvement des objets qui nous entourent peut fournir des informations sur la partie de la scène sur laquelle nous devons nous concentrer, ou sur les objets que nous devons éviter.

Deuxièmement, la détection des mouvements saillants a des applications en matière de sécurité, où elle peut être utilisée, par exemple, pour détecter des mouvements anormaux dans un groupe de personnes ou de voitures dans une vidéo aérienne d'une ville. Dans ces cas, le mouvement joue un rôle critique car l'apparence ne révèle pas la situation anormale. En biologie, la détection d'objets qui ont un mouvement singulier est également intéressante car elle peut permettre de détecter des réactions anormales.

Troisièmement, développer des descripteurs pour représenter le mouvement est également un objectif intéressant car il est à la base de nombreuses tâches en aval. Par exemple, certaines recherches cliniques exploitent les descripteurs de mouvement pour aider à diagnostiquer la maladie de Parkinson à partir de vidéos de mouvements de la main [2]. Nous pouvons également utiliser les descripteurs de mouvement pour améliorer la reconnaissance d'activités ou aider à construire de meilleures interfaces personne-machine. Développer de bons descripteurs de mouvement permet de "capturer" un mouvement à partir d'une vidéo et de le faire correspondre à un mouvement observé dans d'autres vidéos.

Enfin, une autre motivation, qui est au cœur de plusieurs travaux récents [3]–[5] est d'apprendre des descripteurs d'apparence à partir de la segmentation du mouvement. L'idée repose sur le principe de Gestalt et sur le fait que des points se déplaçant solidairement sont suceptibles d'appartenir au même objet. Par conséquent, en faisant coïncider la segmentation basée sur l'apparence avec la segmentation basée sur le mouvement, nous pouvons apprendre les descripteurs d'apparence des objets sans dépendre des annotations humaines, qui sont coûteuses à obtenir. Cela conduit à la perspective passionnante que nous pourrions apprendre à segmenter des objets dans des images statiques simplement en regardant des vidéos.

Développement de descripteurs de mouvement

.

Le domaine de l'apprentissage profond a récemment développé de très bonnes représentations pour les images [6], le son [7], le texte [8] et montrent qu'il permet des applications impressionnantes et facilitent l'apprentissage des tâches en aval. Des descripteurs de vidéo informatifs ont également été développées pour décrire les vidéos avec par exemple des travaux sur la reconnaissance d'activités [9] où le réseau est entraîné à classer les vidéos dans un ensemble fini d'activités prédéterminées et doit donc apprendre à représenter le clip vidéo de manière compacte. La plupart de ces travaux s'appliquent au niveau de la vidéo et prennent en compte l'apparence, qui peut être un discriminant fort dans certaines activités. En outre, comme ils sont élaborés sur un ensemble forcément restreint de vidéos, il y a une limite en termes de généralisation. Le développement de descripteurs de mouvement généraux pouvant être utilisés dans n'importe quel contexte constitue un objectif intéressant pour faciliter l'analyse des mouvements. Cela doit permettre de comparer deux mouvements différents ou de faire correspondre un mouvement observé à une base de données de mouvements connus.

L'élaboration de descripteurs à partir de mouvement n'est pas nouvelle et n'est pas non plus propre à l'apprentissage profond. Par exemple, la notation Laban illustée à la Fig. 1 a été développée en 1928 pour représenter le mouvement des corps humains dans les chorégraphies de danse. Ce système utilise des symboles pour décrire la direction du mouvement dans l'espace, la partie du corps qui exécute le mouvement (chaque partie du corps humain est étiquetée avec un symbole différent), la durée du mouvement (chaque ligne représente un pas de temps, et les mouvements simultanés sont représentés sur la même ligne) et la dynamique du mouvement. Bien qu'il existe de nombreuses notations de mouvement comme celle-ci, l'apprentissage profond pourrait aider à construire des représentations plus générales qui pourraient s'appliquer non seulement au mouvement humain, mais aussi à tous les mouvements qui nous entourent. Comme nous l'avons vu précédemment, cette notation pourrait être utile dans plusieurs types de tâches.

Principaux objectifs de l'analyse du mouvement

Le flot optique est défini comme le champ de déplacement 2D qui décrit le mouvement apparent des pixels entre deux images successives [10]. Les approches classiques s'appuient généralement sur l'hypothèse de conservation de l'intensité (Brightness Con-

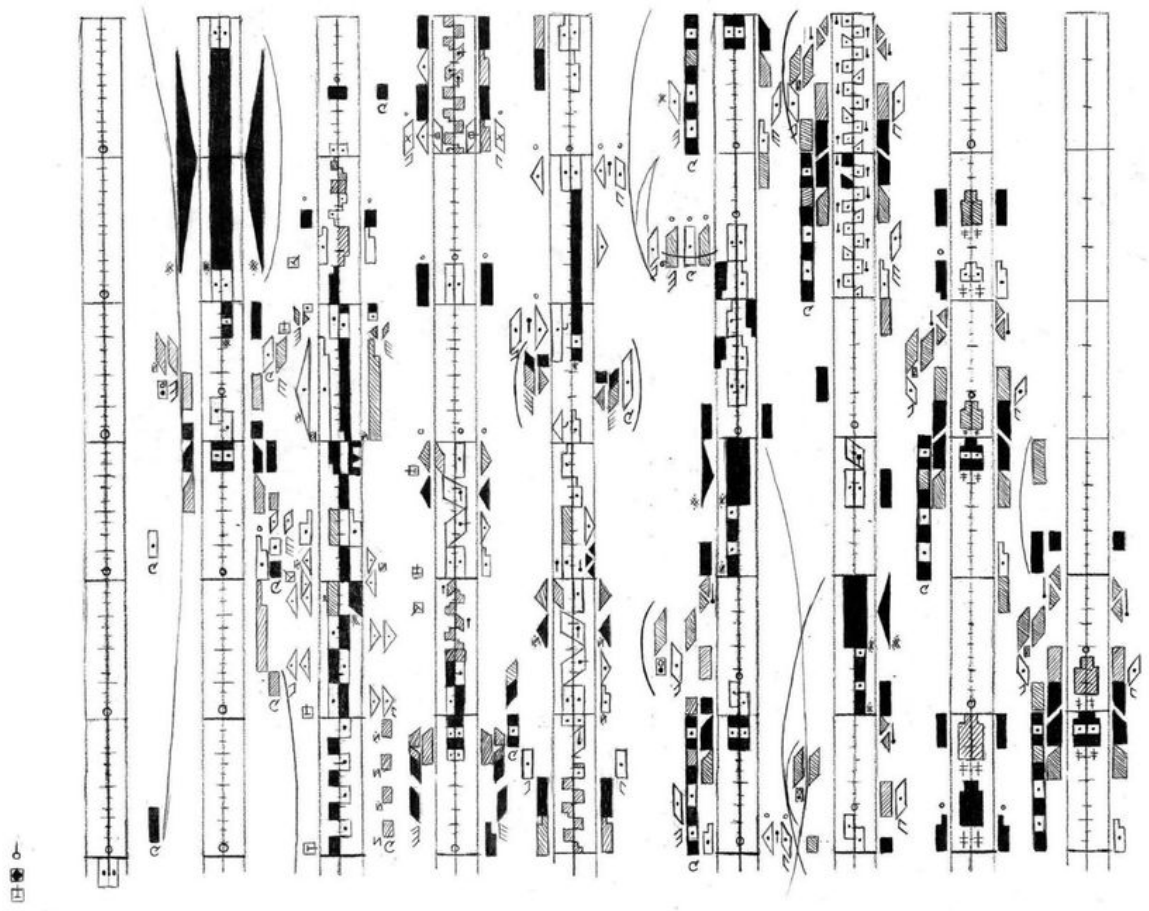


Figure 1 – Exemple de partition utilisant le système de notation Laban. - Loren Foster, France Culture - "Danse : comment s'écrit le mouvement ?"

stancy Equation -BCA- en anglais), qui suppose que l'intensité d'un pixel en mouvement reste constante dans le temps. Plus récemment, les approches par apprentissage profond ont atteint les meilleurs résultats dans l'estimation du flot optique. La plupart de ces méthodes sont entraînées dans un cadre supervisé sur une base de données synthétiques. Toutefois, certaines méthodes sont entraînées dans un cadre non supervisé en utilisant des fonctions de pertes s'appuyant sur les fonctionnelles précédemment définies dans un cadre variationnel.

Les techniques de segmentation du mouvement visent à distinguer et séparer les mouvements 2D au sein d'une vidéo, quelle que soit la source de chaque mouvement individuel. Aujourd'hui, il est possible de réaliser la segmentation du mouvement directement à partir d'un champ de flot optique, et de développer des approches d'apprentissage profond non supervisées, qui apprennent la segmentation du mouvement sans annotations humaines. Nous proposons une taxonomie des techniques de segmentation du mouvement à la figure 1.1 du chapitre 1, où nous soulignons que nous pouvons diviser les méthodes en correspondance visuelles intra-video, en regroupement de trajectoires et en segmentation du mouvement instantané. Cette dernière catégorie est au cœur de notre travail.

La segmentation des objets mobiles dans les vidéo (Video Object Segmentation -VOS- en anglais) est l'une des principales tâches en analyse du mouvement. L'objectif est de segmenter dans la vidéo l'objet principal en mouvement au premier plan de la scène. Les méthodes de VOS sont évaluées sur des ensembles de données tels que DAVIS2016. Dans la figure 2, nous montrons deux séquences de DAVIS2016 où nous pouvons voir que l'objet segmenté est l'objet principal en mouvement dans l'ensemble des images de la séquence. La plupart des méthodes de VOS combinent des informations d'apparence et des informations de mouvement obtenues via le champ de flot optique. En outre, nous pouvons également mentionner des méthodes s'attaquant à la segmentation amodale, une tâche connexe où l'objectif est de segmenter les objets dans leur ensemble même lorsqu'ils sont partiellement occultés ou lorsqu'ils s'arrêtent. Cette tâche est plus difficile car elle nécessite de modéliser la forme de l'objet pour la prédire lorsqu'elle est occultée. Un autre intérêt de l'étude des méthodes VOS est qu'elles donnent une idée de ce que les méthodes non supervisées peuvent réaliser si nous trouvons les fonctions de perte et les procédures d'apprentissage adéquates.

Enfin, nous pouvons mettre en avant la saillance des mouvements (Motion Saliency -MS- en anglais), qui consiste à mettre en évidence les mouvements locaux qui se distinguent de leur contexte environnant et tendent ainsi à révéler un événement important.

Figure 2 – Deux séquences du dataset DAVIS2016 avec la vérité-terrain surimposée en jaune (objet mobile en avant-plan).

Motivations

De nombreux travaux ont été réalisés dans le domaine de la segmentation du mouvements et les progrès ont été rapides ces dernières années. Cependant, il reste encore des défis à relever dans cette tâche. Nous soulignons ci-dessous les motivations qui ont guidé notre travail et notre méthodologie en analyse du mouvement.

Dans de nombreux travaux sur l'analyse de vidéo, l'apparence et le mouvement sont combinés. Plusieurs travaux montrent que cette fusion permet d'obtenir des améliorations significatives sur les benchmarks classiques de segmentation VOS, ce qui justifie l'adoption de caractéristiques d'images. Cependant, il existe un biais important dans ces ensembles de données : l'objet à détecter n'est pas seulement saillant par son mouvement par rapport à l'arrière-plan, mais aussi par son apparence et son emplacement, l'objet étant généralement placé au centre des images (ce qui est un indice explicitement exploité par plusieurs approches [11], [12]).

Le fait que la plupart des travaux récents sur l'analyse de vidéo combinent les deux modalités pourrait être un problème, car il limite l'utilisation de ces algorithmes aux vidéos pour lesquelles l'apparence de l'objet d'intérêt peut être facilement représentée par un réseau neuronal. Par exemple, de nombreuses méthodes reposent sur l'utilisation de caractéristiques DINO qui ont été entraînées sur un vaste ensemble de vidéos naturelles provenant d'Internet, ce qui ne peut pas être appliqué à d'autres types de vidéos représentant des objets pour lesquels nous ne disposons pas d'une telle base de données, comme les images satellites, les images biologiques, les vidéos sous-marines ou industrielles. Par exemple, les résultats expérimentaux en apprentissage profond montrent que les modèles

entraînés sur des flots optiques, y compris synthétiques, ont une bien meilleure capacité de généralisation que ceux entraînés sur les images. En outre, le fait de se concentrer uniquement sur le mouvement permet des transferts intéressants qui modifient le contexte des vidéos. On pourrait imaginer transférer un algorithme de détection de saillance du mouvement entraîné sur des foules à l'analyse du mouvement de groupes de cellules.

De même, l'entraînement supervisé limite l'applicabilité des méthodes. Premièrement, elle nécessite une grande quantité de données annotées par des humains, ce qui est coûteux à obtenir. Deuxièmement, il contraint la tâche et peut introduire des biais. Par exemple, si nous prenons la tâche de segmentation du mouvement, toutes les bases de données segmentent l'objet principal (ou les quelques objets principaux pour FBMS59 et DAVIS2017) dans la scène. Cependant, dans la pratique, lorsque nous regardons ces vidéos, nous pouvons y trouver d'autres mouvements à segmenter qui ne sont pas recensés dans la vérité terrain. Tout d'abord, nous avons des mouvements indépendants d'objets d'arrière-plan comme dans les séquences *breakdance* ou *scooter-black* de DAVIS2016, des mouvements dus à l'eau comme dans la séquence *blackswan* de DAVIS2016. Dans un environnement naturel, on pourrait également évoquer le mouvement des arbres ou des nuages lorsqu'il y a du vent. Deuxièmement, nous avons la parallaxe du mouvement liée aux objets statiques d'avant plan et causée par le mouvement de la caméra et les différences de profondeur. Elle peut entraîner de fortes différences de flot, comme dans les séquences de *parkour* ou de *libby* dans DAVIS2016. Troisièmement, nous avons les mouvements articulés, qui pourraient être traités de manière plus complète. Tous ces mouvements nous donnent des informations précieuses sur le monde qui nous entoure et sont écartés dans les approches supervisées, car il serait impossible de tous les segmenter manuellement. Par conséquent, l'apprentissage de la segmentation des mouvements de manière supervisée ne résout qu'une partie du problème. Nous pensons que pour aborder la segmentation du mouvement, la bonne approche est l'apprentissage non supervisé. C'est la seule qui puisse nous conduire à une compréhension complète du mouvement de la scène.

Principales contributions

Pour les raisons susmentionnées, nous avons choisi d'établir un cadre bien formalisé pour ce travail. Nous essayons d'identifier des objets mobiles saillants ou de segmenter des objets mobiles indépendants en nous basant uniquement sur leur mouvement. Cela

implique un certain nombre de décisions méthodologiques. Tout d'abord, nous ne nous appuyons pas sur l'apparence ou sur des informations sémantiques. Deuxièmement, nous n'avons pas besoins des informations d'égomotion ou des paramètres intrinsèques de la caméra. Enfin, notre procédure d'apprentissage ne repose pas sur l'annotation manuelle. Cela nous conduit au défi stimulant d'exploiter les seuls champs de flot optique qui indiquent seulement le mouvement des points individuels d'une paire d'images.

Les contributions de cette thèse s'articulent autour de deux grands axes d'analyse du mouvement.

Le premier concerne les approches d'apprentissage profond pour la segmentation non supervisée du mouvement. Dans cet axe, nous construisons un cadre pour entraîner un réseau à segmenter le mouvement à partir du champ de flot optique en utilisant une fonction de perte inspirée de l'algorithme EM. Il décompose le flot en segments de mouvement cohérents, chacun représenté par un modèle de mouvement paramétrique. Nous étendons ensuite progressivement ce cadre à des séquences vidéo plus longues. Dans un premier travail, nous prenons en entrée des triplets de champs de flot optique. Nous introduisons la dimension temporelle à court terme avec un terme de la fonction de perte qui assure la cohérence des étiquettes au sein des triplets. Nous l'étendons à une cohérence temporelle à long terme par un post-traitement qui relie les triplets pour former une segmentation longue et temporellement cohérente. Dans un second travail, nous introduisons une représentation du mouvement basée sur les splines qui est capable de représenter l'évolution d'un champ de mouvement paramétrique dans le temps. Nous considérons sur un réseau neuronal incluant des transformers pour permettre des interactions entre les caractéristiques de la séquence complète. Il en résulte une segmentation du mouvement temporellement cohérente sur l'ensemble de la séquence, sans avoir recours à un quelconque post-traitement. Toutes ces méthodes ont été testées sur des benchmarks VOS classiques (DAVIS2016, FBMS59, SegTrackV2) et fournissent des résultats compétitifs tout en étant efficaces au moment du test.

Le deuxième axe est la localisation des mouvements saillants à partir des champ de flot optique. Dans cette partie, nous articulons notre travail autour de l'idée que les zones saillantes sont celles qui influencent la sortie d'un réseau pré-entraîné qui prédit si un champ de flot optique comprend des mouvements saillants. Dans cette définition, la saillance est spécifique à la tâche que le réseau tente de résoudre. Dans un premier travail, nous utilisons une méthode d'interprétation de réseau par gradient pour localiser les zones saillantes. Nous y adjoignons une méthode de segmentation du mouvement.

Dans un second travail, nous optons pour une approche adverse afin de localiser les zones saillantes. Nous substituons le flot optique par inpainting dans le but de modifier la sortie du réseau de classification. Nous avons considéré deux tâches différentes de saillance de mouvement, l'une impliquant la détection de mouvements indépendants dans une scène observée par une caméra en mouvement, et l'autre impliquant la détection de mouvements anormaux de piétons dans une foule.

Organisation du manuscrit

Ce manuscrit comporte deux parties principales après le chapitre sur la description de l'état de l'art sur la segmentation du mouvement.

La première partie, composée des chapitres 2, 3 et 4, se concentre sur la segmentation du flot optique non supervisée à l'aide de réseaux profonds. Dans le chapitre 2, nous décrivons notre méthode de segmentation de mouvement du flot optique calculé entre deux images consécutives. Le chapitre 3 détaille notre extension à la segmentation de petits volumes de flot optique, fournissant une segmentation temporellement cohérente et une robustesse aux valeurs aberrantes. Le chapitre 4 décrit l'introduction de splines pour les modèles de mouvement et les transformers pour une segmentation temporellement cohérente de volumes de flot optique plus conséquents.

La deuxième partie, composée des chapitres 5 et 6, se concentre sur la détection faiblement supervisée des mouvements saillants dans les champs de flot optiques. Dans le chapitre 5, nous décrivons notre méthode de localisation des mouvements saillants à l'aide d'une interprétation par le gradient d'un réseau de classification. Dans le chapitre 6, nous présentons une méthode d'entraînement d'un réseau de segmentation pour localiser les zones saillantes à l'aide d'une approche adverse où nous masquons sélectivement une partie du champ de flot optique d'entrée.

Nous donnons plus d'informations sur chaque chapitre dans ce qui suit.

Chapitre 1

Dans le chapitre 1, nous présentons les travaux connexes principaux, en nous concentrant sur la segmentation du mouvement, mais aussi sur la segmentation des objets moviles dans des vidéos (VOS) et la détection de mouvements saillants. Nous présentons d'abord les méthodes traitant de la segmentation VOS. Ensuite, nous nous concentrons

sur les méthodes liées à la segmentation du mouvement, en décrivant une taxonomie des méthodes. Enfin, nous présentons les méthodes liées à la saillance du mouvement, en traitant également les méthodes de saillance vidéo basées sur les caractéristiques d'apparence et de mouvement et les méthodes de saillance basées sur l'eye-tracking.

Chapitre 2

Dans le chapitre 2, nous définissons une méthode par apprentissage profond entièrement non supervisée pour segmenter les champs de flot optique en régions de mouvement cohérent. Nous supposons que le flot optique d'entrée peut être représenté comme un ensemble de modèles polynomiaux, typiquement des modèles de mouvement affine ou quadratique, chacun caractérisant un segment de mouvement (un segment est soit une région, c'est-à-dire une composante connectée, soit une couche qui n'est pas nécessairement connectée). L'idée centrale de notre travail est de se fonder sur l'approche EM (Expectation-Maximisation). Cela nous permet de concevoir de manière mathématiquement bien fondée la fonction de perte et la procédure d'entraînement de notre réseau neuronal de segmentation du mouvement. Contrairement à la méthode EM classique, notre réseau, une fois entraîné, peut fournir une segmentation pour n'importe quel champ de flot optique non vu en une seule étape d'inférence, sans dépendre de l'initialisation et sans estimer de modèles de mouvement paramétriques. Différentes fonctions de perte ont été étudiées, y compris des fonctions robustes. Nous avons également défini un nouveau schéma d'augmentation des données dédié au flot optique, qui a un impact notable sur la performance du réseau. Notre réseau de segmentation du mouvement a été testé sur quatre benchmarks, DAVIS2016, SegTrackV2, FBMS59, et MoCA, et a obtenu de très bons résultats tout en étant rapide au moment du test. Le logiciel EM-Flow qui met en œuvre cette méthode est disponible sur Github (https://github.com/Etienne-Meunier-Inria/EM-Flow-Segmentation).

Chapitre 3

Dans le chapitre 3, nous définissons une méthode de segmentation pour les séquences de champs de flot optique, en introduisant la cohérence temporelle à notre travail développé précédemment. La cohérence temporelle est une caractéristique clé dans la segmentation de mouvement, mais elle est souvent négligée dans les méthodes non supervisées de segmentation de flot optique. Nous avons proposé un cadre spatio-temporel non supervisé

original pour la segmentation du mouvement par flot optique qui explore pleinement la dimension temporelle du problème. Plus précisément, nous avons défini un réseau 3D pour la segmentation de mouvements multiples qui prend en entrée un sous-volume de flots optiques successifs et renvoie un sous-volume correspondant de cartes de segmentation cohérentes. Notre réseau est entraîné de manière totalement non supervisée. La fonction de perte combine un terme de reconstruction de flot qui incorpore des modèles de mouvement paramétriques spatio-temporels, et un terme de régularisation qui renforce la cohérence temporelle des masques. Nous avons spécifié un post-traitement simple pour la liaison temporelle à long terme des segments prédits, ce qui permet d'obtenir une segmentation cohérente sur de longues séquences tout en ne donnant que des triplets au réseau de segmentation. En outre, nous avons conçu une méthode flexible et efficace pour encoder les réseaux Unets. Nous avons réalisé des expériences sur plusieurs datasets avec des résultats quantitatifs convaincants. L'implémentation logicielle appelée ST-Segmentation est disponible sur github (https://github.com/Etienne-Meunier-Inria/ST-Segmentation).

Chapitre 4

Dans le chapitre 4, nous présentons les modèles de mouvement s'appuyant sur des splines pour une représentation du mouvement adaptée à une longue période de temps. Nous montrons que l'avantage de ces modèles de mouvement est qu'ils peuvent représenter une évolution temporelle significative des modèles de mouvement paramétriques. Nous introduisons également une architecture de type maskformer [13] pour segmenter le flot optique et un module d'attention temporelle qui permet l'interaction entre les caractéristiques sur l'ensemble de la séquence. Nous montrons que la combinaison de ces deux contributions permet d'entraîner un réseau capable de segmenter de longues séquences de flot optique de manière non supervisée. Notre réseau de segmentation de mouvement a été testé sur quatre benchmarks, DAVIS2016, DAVIS2017, SegTrackV2, FBMS59, et montre des résultats compétitifs, fournissant une segmentation précise tout en maintenant la cohérence de l'étiquette à long terme. Il est également très rapide, capable de traiter des séquences de plusieurs centaines d'images en un seul passage à une moyenne de 210 fps. L'implémentation logicielle appelée LT-Segmentation est disponible sur github (https://github.com/Etienne-Meunier-Inria/LT-Segmentation).

Chapitre 5

Dans le chapitre 5, nous introduisons un nouveau paradigme pour calculer le mouvement saillant dans les images vidéo à partir uniquement du flot optique. La saillance du mouvement (MS) est une question importante dans l'analyse de scènes dynamiques, quel que soit le domaine d'application (par exemple, l'observation au microscope de cellules vivantes, la télédétection ou la navigation autonome). La saillance des mouvements vise à mettre en évidence les mouvements locaux qui s'écartent du contexte environnant, ce qui tend à révéler des événements dynamiques significatifs. Nous avons formulé la saillance du mouvement comme une méta-tâche qui peut être instanciée pour différentes tâches habituellement traitées de manière indépendante. Pour étayer cette affirmation, nous avons abordé deux problèmes généraux importants : la segmentation d'objets mobiles dans une scène observée avec une caméra en mouvement et la détection de mouvements anormaux dans des foules.

Nous estimons la saillance du mouvement à partir de l'interprétation d'un réseau de classification de la saillance du mouvement prenant des flots optiques en entrée. Notre paradigme peut s'adapter à toute forme de saillance du mouvement en entraînant simplement le réseau de classification sur la tâche appropriée. Notre estimation de la saillance des mouvements est non supervisée. Toutefois la classification initiale est faiblement supervisée. Nous avons développé un schéma original d'interprétation du réseau en deux étapes exploitant la méthode LRP. Nous avons besoin de lui adjoindre une technique de segmentation de flot optique. Il fournit une segmentation binaire des mouvements saillants. Enfin, nous pouvons récupérer la carte de saillance de mouvement valuée par inpainting de flot paramétrique. Nous testons notre méthode sur deux cas de saillance différents. Le premier est le mouvement saillant produit par des objets se déplaçant indépendamment les uns des autres dans une scène observée par une caméra en mouvement. Le second cas est le mouvement saillant résultant d'un mouvement distinctif au sein d'un ensemble de mouvements cohérents. Les résultats expérimentaux sur des vidéos réelles et la comparaison avec les méthodes existantes démontrent la performance de notre méthode.

Chapitre 6

Dans le chapitre 6, nous présentons une première étude d'une approche adverse pour apprendre à segmenter les régions de mouvement saillant dans les champs de flot optique, à partir de la prédiction d'un réseau de classification. Dans ce travail, nous fusionnons

les étapes de segmentation et d'interprétation pour produire directement la segmentation du mouvement saillant à partir du champ de flot optique. Au lieu de s'appuyer sur des techniques d'interprétation du flot, telles que la LRP, nous formons un réseau de segmentation pour localiser les zones qui, lorsqu'elles sont modifiées par inpainting paramétrique, changent la prédiction du réseau de classification. Nous montrons également que nous pouvons le formuler comme une tâche d'optimisation sous contrainte et utiliser une approche basée sur le lagrangien pour calculer automatiquement un hyperparamètre critique, la pondération entre les termes de la fonction de perte de notre réseau. A ce stade, nous montrerons des résultats préliminaires essentiellement sur la tâche de détection du mouvement saillant correspondant à un mouvement distinctif au sein d'un ensemble de mouvements cohérents.

General Introduction

Context and Motivations

Context of motion analysis

The way we perceive and interpret motion provides us with critical information about our environment. From a functional perspective, because a moving object may represent something particularly worthy of attention, motion helps us focus on what is important around us. Motion also embeds the fact that the world around us is changing and that we need to pay attention to this updated version of our environment. Much like appearance, the motion of the objects or entities around us can be subject to interpretation and provide us with useful cues about how to interact with them.

In the context of research trying to develop AI systems to assist humans, it is crucial to build mechanisms that are able to perceive and represent the surrounding motion in a meaningful way. However, this task is complicated to reproduce because the human visual system is surprisingly good at understanding motion. For example, when walking in a crowded place with many different motions, we are able to quickly notice a single one among them and focus on it. Furthermore, when running in a forest where we are able to ignore the motion parallax of the trees around us in order to focus on the potential presence of independently moving objects. A review of this question and some explanations on how the human brain handles this task is given in previous works [1].

Practical motivations for motion analysis :

Motion analysis is one of the main tasks in computer vision and has many applications in a wide variety of fields. First, segmentation of moving objects has applications in video editing, where we can select objects based on their motion, but also in robotics or autonomous driving, where the motion of objects around us can provide information about which part of the scene we should focus on, or about objects we have to avoid.

Second, the detection of salient motion has security applications, where it can be used, for example, to detect abnormal motion in a group of people or of cars in an aerial video

of a city. In these cases, motion plays a critical role because the appearance does not reveal the abnormal situation. In biology, the detection of objects that have a singular motion is also of interest because it can allow to detect abnormal reactions.

Third, developing features to represent motion is also an interesting task as it is the foundation of many downstream tasks. For example, some projects exploit motion descriptors to help to diagnose Parkinson's disease from hand movement videos [2]. We could also use motion descriptors to improve activity recognition or help to build better human-machine interfaces. Developing good motion description features could allow us to "capture" a motion from a video and match it with motion observed in other videos.

Finally, another motivation, which is at the core of several recent works [3]–[5] is to learn appearance descriptors from motion segmentation. The idea is based on the Gestalt principle and the fact that points that move together belong to the same object. Therefore, by making appearance-based segmentation coincide with motion-based segmentation, we can learn object features without relying on human annotations, which are expensive to obtain. This leads to the exciting perspective that we could learn to segment objects in static images just by watching videos.

Development of general motion features

Deep learning (DL) area recently developed very good feature embeddings for images [6], sound [7], text [8] and show that it allows impressive applications and facilitate learning downstream tasks. Informative video features have also been developed to describe videos with for example works on activity recognition [9] where the network is trained to classify videos into a finite set of predetermined activities and thus must learn to represent the video clip in a compact way. Although most of these works proceed at the video level and take into account the appearance, which can be a strong discriminator in certain activities, furthermore as they are trained on a limited set of videos, there is a limit in terms of generalization. The development of general motion descriptors that can be used in any context seems to be an interesting possibility to facilitate downstream motion analysis. It would allow one to compare two different motions or to match an observed motion against a database of known motions.

Developing descriptors from motion is not new, nor is it unique to deep learning. For example, the Laban notation was developed in 1928 to represent the motion of human bodies in dance choreography (see Fig.3). This system uses symbols to describe the direction of the movement in space, the part of the body performing the movement (each

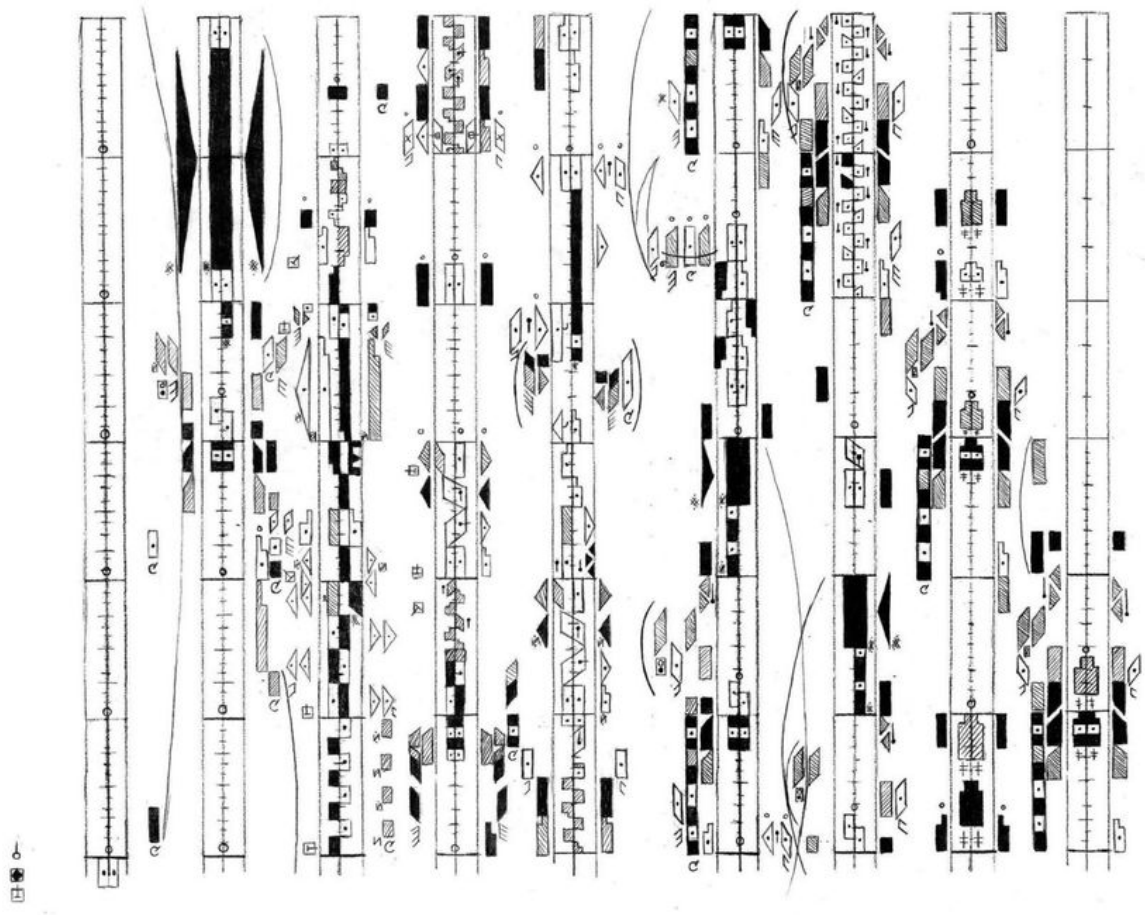


Figure 3 – Example of a partition using the Laban notation system. - Loren Foster, France Culture - "Danse : comment s'écrit le mouvement ?"

part of the human body is labeled with a different symbol), the duration of the movement (each line represents a time step, and simultaneous movements are represented on the same line), and the dynamics of the movement. Although many motion notations like this exist, deep learning could help build more general representations that could apply not only to human motion, but to all motion around us. As we discussed earlier, this notation could help in several downstream tasks.

Main objectives of motion analysis

Optical flow is defined as the 2D displacement field that describes the apparent motion of pixels between two successive images [10]. Classical approaches typically rely on the

Brightness Constancy Assumption (BCA), which assumes that the brightness of a moving pixel remains constant over time. More recently, deep learning approaches have shown the best results in estimating optical flow. Most of these methods are trained in a supervised setting on a synthetic database, and some methods are trained in an unsupervised setting using losses based on image warping.

Motion segmentation techniques aim to partition 2D motion along a video, regardless of the source of each individual motion. Nowadays, one can rely on optimizing the segmentation directly on a given optical flow field, and develop unsupervised deep learning approaches, that learn to segment unseen optical flow fields without relying on human annotation. We propose a taxonomy of motion segmentation techniques in Figure 1.1 of Chapter 1, where we highlight that we can divide methods into intra-video correspondence, trajectory clustering, and instantaneous motion segmentation. This latter category is central to our work.

Video Object Segmentation (VOS) is one of the prominent task in motion analysis. The objective is to segment throughout the video the primary moving object in the foreground of the scene. The VOS methods are evaluated on datasets such as DAVIS2016. In Fig.4, we show two sequences from DAVIS2016 where we can see that the segmented object is the principal moving object common to all frames in the sequence. Most of the VOS methods combine appearance information and motion information obtained via the optical flow field. Furthermore, we can also mention methods tackling amodal segmentation, a related task where the goal is to segment objects as a whole even when they are partially occluded or when they stop. This task is more challenging as it requires modeling the object shape to predict it when it is occluded. Another interest in studying these methods is that they give an idea of what unsupervised methods can achieve if we find the right loss functions and training procedures.

Finally, we can put forward motion saliency (MS), which is the task of highlighting local motions that deviate from their surrounding context and thus tend to reveal a significant event.

Motivations

Many works have been carried out in the field of motion segmentation and the progress has been really fast these last years. However, there are still challenges to overcome in this task. In this subsection, we highlight the rationale that guided our work and methodology

Figure 4 – Two sequences of DAVIS2016 dataset with ground-truth segmentation mask overlayed in yellow.

on motion analysis.

In many works on video analysis, appearance and motion are combined. Several works show that this fusion yields significant improvement on classical video object segmentation benchmarks, justifying the adoption of image features. However, there is an important bias in these datasets: the object to be detected is not only salient by its motion compared to the background, but also by its appearance and its location, the object being generally placed in the center of the images (which is a cue explicitly exploited by several approaches [11], [12]).

The fact that most recent works on video analysis combine the two modalities could be a problem, as it limits the use of these algorithms to videos for which the appearance of the object of interest can be easily represented by a neural network. For example, many methods rely on the use of DINO features that have been trained on a large set of natural videos from the Internet, which cannot be applied to other types of videos that represent objects for which we do not have such a database, such as satellite images, biological images, underwater or industrial videos. For example, experimental results in deep learning show that models trained on optical flow have a much better generalization ability than those trained on images. Moreover, focusing only on motion allows interesting transfers that change the context of the videos. We could imagine transferring a motion saliency detection algorithm trained on crowds to the analysis of the motion of groups of cells.

Similarly, supervised training limits the applicability of the methods. First, it requires a large amount of human annotated data, which is expensive to obtain. Second,

it constrains the task and can introduce biases. For example, if we take the task of motion segmentation, all databases segment the main object (or the few main objects for FBMS59 and DAVIS2017) in the scene. However, in practice, when we look at a video, we may have many motions to segment that are not segmented in the ground truth. First, we have independent motions of background objects as in the *breakdance* or *scooter-black* videos in DAVIS2016, motion due to water as in the *blackswann* video in DAVIS2016. In a natural environment, we could also mention the movement of trees or clouds when there is wind. Second, we have motion parallax attached to static objects, which is caused by camera movement and depth differences. It can lead to strong flow differences as in the *parkour* or *libby* videos in DAVIS2016. Third, we have articulated motion, which could be handled in a more comprehensive way. All these motions give us valuable information about the world around us and are discarded in supervised approaches, because it would be impossible to manually segment them all. Therefore, training motion segmentation in a supervised way only solves part of the problem. We believe that to address motion segmentation, the good approach is unsupervised learning. It is the only one that can lead us to a holistic understanding of the scene motion.

Main Contributions

For the above reasons, we choose to set up a strict framework for this work. We try to identify salient moving objects or segment independently moving objects based only on their motion. This involves a number of methodological decisions. First, we do not rely on appearance or semantic information. Second, we do not take in account given egomotion information or camera intrinsic parameters. Third, our training procedure does not rely on human annotation. This leaves us with the interesting challenge of analyzing optical flow fields that only indicate the motion of individual points of a pair of images.

The contributions of this thesis are organized around two axes of motion analysis.

The first one is concerned with deep learning approaches for unsupervised motion segmentation. In this axis, we build a framework to train a network to segment motion from an optical flow field using a loss function based on the EM algorithm. It decomposes the flow into coherent motion segments, each represented by a parametric motion model. We then gradually extend this framework to longer video sequences. In a first work, we take as input triplets of optical flow fields, representing a total of four frames in the input video. It introduces short-term temporal consistency with a loss term that

enforces that labels within the triplets are consistent, and long-term temporal consistency with a post-processing that links triplets together to form a long, temporally consistent segmentation. In a second work, we introduce a spline-based motion representation that is able to represent the evolution of a parametric motion field over time, and we rely on a transformer-based neural network to allow interactions between features of the full sequence. This results in a temporally consistent motion segmentation over the entire sequence in one shot without relying on any post-processing. All these methods have been tested on classical VOS benchmarks (DAVIS2016, FBMS59, SegTrackV2, DAVIS2017), and provide competitive results while being very efficient at test time.

The second axis is the localization of salient motions from the optic flow field. In this part, we articulate our work around the idea that salients areas are those that influence the output of a pre-trained network predicting whether an optic flow field includes salient motions. In this definition, saliency is specific to the task the network is trying to solve. In a first work, we use a gradient-based network interpretation method to localize salient areas, and use the flow to improve the provided segmentation. In a second work, we aim to design an adversarial framework to localize salient areas and inpaint them into the input optical flow, with the goal of altering the output of the classification network. We have considered two different motion saliency tasks, one involving the detection of independent motions in a scene observed by a moving camera, and the other involving the detection of anomalous pedestrian motions in a crowd.

Organisation of the manuscript

There are two main parts to this manuscript after the related work chapter.

The first part, consisting of Chapters 2, 3 and 4, focuses on unsupervised optical flow segmentation using deep networks. In Chapter 2, we describe our method for motion segmentation of the optical flow computed between two consecutive frames. Chapter 3 details our extension to the segmentation of small optical flow volumes, providing temporally consistent segmentation and robustness to outliers. Chapter 4 details the introduction of splines and transformer-based models for temporally consistent segmentation of longer optical flow volumes.

The second part, consisting of Chapters 5 and 6, focuses on the weakly supervised detection of salient motion in optical flow fields. In Chapter 5, we describe our method to localize salient motion using a gradient-based interpretation of a classification network.

In Chapter 6, we present a method for training a segmentation network to localize salient areas using an adversarial approach where we selectively mask part of the input optical flow field.

Finally, we have also investigated the optical flow estimation issue. More specifically, we report in Appendix 7.4 an additional work on the estimation of optical flow for small moving objects in large-scale images, which raises specific problems for optical flow methods based on deep learning.

We give more information on each chapter in the following.

Chapter 1

In Chapter 1. we present related work, focusing primarily on motion segmentation, but also on video object segmentation and motion saliency detection. First, we present methods dealing with video object segmentation. Then, we focus on methods related to motion segmentation, describing a taxonomy of methods where we distinguish methods based on trajectory methods and one based directly on instantaneous motion segmentation. Finally, we introduce methods related to motion saliency, treating both video saliency methods based on appearance and motion features and eye-tracking based saliency methods that learn to imitate the human gaze.

Chapter 2

In Chapter 2, we define a fully unsupervised CNN-based method for segmenting optical flow fields into coherent motion regions. We assume that the input optical flow can be represented as a piecewise set of polynomial models, typically affine or quadratic motion models, each of which characterizing a motion segment (a segment is either a region, i.e., a connected component, or a layer that is not necessarily connected). The core idea of our work is to use the Expectation-Maximization (EM) framework. This allows us to design the loss function and training procedure of our motion segmentation neural network in a well-founded way. In contrast to classical EM, our network, once trained, can provide segmentation for any unseen optic flow field in a single inference step, without dependence on initialization and without estimating any parametric motion models. Different loss functions have been investigated, including robust ones. We have also defined a novel data augmentation scheme dedicated to optical flow, which has a noticeable impact on the network performance. Our motion segmentation network has

been tested on four benchmarks, DAVIS2016, SegTrackV2, FBMS59, and MoCA, and performed very well while being fast at test time. The EM-Flow software that implements this method is publicly available on Github (https://github.com/Etienne-Meunier-Inria/EM-Flow-Segmentation).

Chapter 3

In Chapter 3, we define a segmentation method for sequences of optical flow fields, introducing temporal consistency to our previously developed work. Temporal consistency is a key feature in motion segmentation, but it is often neglected in unsupervised optical flow segmentation methods. We have proposed an original unsupervised spatio-temporal framework for optical flow motion segmentation that fully explores the temporal dimension of the problem. More specifically, we have defined a 3D network for multiple motion segmentation that takes as input a sub-volume of successive optical flows and returns a corresponding sub-volume of coherent segmentation maps. Our network is trained in a fully unsupervised manner. The loss function combines a flow reconstruction term that incorporates spatio-temporal parametric motion models, and a regularization term that enforces temporal consistency on the masks. We have specified a simple post-processing for long-term temporal linking of predicted segments allowing to have consistent segmentation along long sequences while giving only triplets to the segmentation network. Furthermore, we have designed a flexible and efficient way to encode U-nets. We have performed experiments on several benchmarks with convincing quantitative results. The software implementation called ST-Segmentation is available on github (https://github.com/Etienne-Meunier-Inria/ST-Segmentation).

Chapter 4

In Chapter 4, we introduce spline-based motion models as a representation of motion over a long period of time. We show that the advantage of these motion models is that they can represent a smooth temporal evolution of the parametric motion models. We also introduce a maskformer type architecture [13] to segment the optical flow and a temporal attention module that enable interaction between features over the entire sequence. We show that the combination of these two contributions allows us to train a network that can segment long optical flow sequences in an unsupervised manner. Our motion segmentation network has been tested on four benchmarks, DAVIS2016, DAVIS2017, SegTrackV2,

FBMS59, and shows competitive results, providing accurate segmentation while maintaining long-term label consistency. It is also very fast, able to process sequences of several hundred frames in one pass at an average of 210 fps. The software implementation called LT-Segmentation is available on github (https://github.com/Etienne-Meunier-Inria/LT-Segmentation).

Chapter 5

In Chapter 5, we introduce a new paradigm for computing salient motion in video frames based solely on optical flow. Motion saliency (MS) is an important issue in dynamic scene analysis, regardless of the application domain (e.g., live cell microscopy, remote sensing, or autonomous navigation). MS aims at highlighting local motions that deviate from their surrounding context, thus tending to reveal significant dynamic events. We have formulated MS as a meta-task that can be instantiated for different tasks that are usually handled independently. To support this claim, we addressed two important general problems with our MS paradigm: the segmentation of independently moving objects and the detection of anomalous motion in videos captured by a mobile camera.

We estimate MS from the interpretation of a frame-based motion saliency classification network with optical flow as input. Our paradigm can accommodate any form of motion saliency by simply training the frame-based classification network on the appropriate task. Our MS estimation is unsupervised as it does not require ground-truth saliency maps for training. We have developed an original two-step network interpretation scheme using the LRP method. It provides the binary salient motion segmentation. Finally, we can recover the valued motion saliency map using parametric flow inpainting. We test our method on two different saliency cases. The first is salient motion produced by independently moving objects in a scene observed by a moving camera. The second case is the salient motion resulting from a distinctive motion within a coherently moving set. Experimental results on real videos and comparison with existing methods demonstrate the performance of our method.

Chapter 6

In Chapter 6, we present a first investigation of an adversarial approach for learning to segment regions of salient motion in optic flow fields based on the prediction of a classification network. The overall goal is to fuse the segmentation and interpretation steps

to directly output the salient motion segmentation from the optic flow field. Instead of relying on gradient-based interpretation techniques such as LRP, we train a segmentation network to locate areas that, when modified (or perturbed), change the prediction of the classification network. We also show that we can formulate it as a constrained optimization task and use a Lagrangian-based approach to automatically compute a critical hyperparameter, the weight between the terms of our network loss. We report preliminary results, mainly on the second motion saliency task, e.g., salient motion resulting from a distinctive motion within a coherently moving set.

Appendix

We have also added several appendices. The three first ones contain supplementary material respectively related to Chapters 2, 3 and 4. The last one corresponds to an additional work on the estimation of optical flow for small moving objects in large-scale images along with some preliminary results.

Chapter 1

Related Work

In this chapter, we review work related to motion segmentation (in a broad sense) in videos. The first part of the chapter focuses on Video Object Segmentation (VOS) that aims to detect the main moving object in videos. The second section introduces more general motion segmentation methods that deal with the partitioning of the 2D motion along a video independently of the source of each motion. We will give a more important focus to this section as it is the most related to our own research. The third section describes methods that deal with the detection of salient motion in videos, including both methods based on motion characteristics and methods that train a neural network to predict human attention from eye tracking data.

1.1 Video Object Segmentation (VOS)

1.1.1 Combining motion and appearance

The focus of Video Object Segmentation (VOS) is on segmenting primary objects (typically a single one) that move in the foreground of a scene and are usually followed by the camera. VOS provides a binary segmentation, primary object versus background [14]. However, the background may also contain moving objects, as in some videos from the DAVIS2016 dataset [15]. The availability of large annotated VOS datasets makes it possible to apply supervised deep learning techniques to VOS.

In the context of several VOS benchmarks, the term "unsupervised" is not used in the same way as in the deep learning community. For these benchmarks, it refers to the fact that no annotation is given at test time, on the contrary with the "semi-supervised" category, where the first frame of each test sequence is annotated with the primary moving object. For the deep learning community, "unsupervised" rather means that no annotated data were used for training, and "supervised" means that annotations were used for training, although in both cases they do not imply the use of annotations for test sequences. In this section, we only mention methods that would be qualified as "unsupervised" in the sense of VOS. Thus, we use the deep learning vocabulary to distinguish between methods trained on annotated data ("supervised") and the other methods ("unsupervised").

In [16], the authors trained a convolutional network to segment optical flow fields. The network is trained in a supervised manner on synthetic data and provides segmentation from the motion field only. This segmentation is later refined using pre-trained appearance features to account for parallax motion and errors in the optical flow field.

Subsequently, several works have shown that the joint use of object appearance and motion improves performance in VOS. In [17], the authors train a two-branch network to predict an optical flow field estimation together with a segmentation from a pair of frames. They show that this setting yields better results than treating the two problems independently as learned features, and that it benefits both tasks. In the same vein, several works combine a pre-trained optical flow with an image to perform VOS segmentation. In [18], the authors train a module that fuses deep features of two convolutional networks, each taking as input one of the modalities, using a custom fusion module that takes into account the confidence of the two different streams. In another paper, MATNet [19] offers an alternative to this fusion process, using an attention-based architecture that allows a better interaction between these two modalities. Finally, in [20], the authors combine the two modalities to build the feature map of a mask R-CNN model used for both segmentation and bounding box prediction.

Recent works have revisited the way of coupling appearance and motion. The AMD method [5] includes two pathways, the appearance one and the motion one, that are then combined to predict segment flow and separate moving objects from background. It does not use optical flow as input and brings out the objectness concept. However, it relies on a coarse motion representation. The RCF method described in [21] involves learnable motion models and is structured in two stages: first, a motion-supervised object discovery stage, then, a refinement stage with residual motion prediction and high-level appearance supervision. However, the method does not impose temporal consistency and cannot distinguish objects undergoing different motions. In a different approach [3], the prediction of probable motion patterns is used at the training stage as a cue to learn objectness from videos.

1.1.2 Exploiting the temporal dimension

It can be beneficial to exploit the intrinsic temporal nature of a video sequences to improve segmentation. In [16], the authors combine the features extracted from convolutional networks applied to the frames and the optical flow fields along a sequence using a recurrent module. The features provided by this recurrent module are then used directly for segmentation. A similar recurrent module is used in [22] which, unlike this previous work, only takes the optical flows as input to focus on motion cues for their segmentation. In [23], the authors provide a framework for segmenting a sequence of images, combining a convolutional network and a recurrent network to take into account features from longer sequences. They focus on the architectural aspect of the network by introducing a multi-scale feature extraction module based on extended convolutions. Another work, [24], trains a co-segmentation module to segment the object common to a pair of frames using a correlation volume. At test time, they use this co-segmentation module to obtain the segmentation for the whole sequence from a set of reference frames, thus, introducing a temporal dimension to their method.

1.1.3 Amodal segmentation

Several papers have introduced the use of supervised training and temporal networks for amodal segmentation, which according to [25] "refers to the task of segmenting the object as whole, including the portions that are partially occluded". In the case of motion segmentation, this may involve cases where an object is occluded or static for a short

period of time. In [25], the authors use a synthetic database to train a transformer model that takes a sequence of optical flow fields as input and returns both amodal and modal segmentation. Another paper [26] uses a similar protocol, although they only predict the amodal segmentation and the ordering between layers, and combine these to form the modal segmentation, taking into account occlusions. Both of these works include a temporal dimension, as they segment a sequence of optical flow fields in one go.

1.1.4 Unsupervised methods

Unsupervised VOS methods have also been developed. In [27], the authors exploit the recurrence property of the primary moving objects to select regions from a set of candidates extracted using colours and motion edges. This recurrence property is based on the fact that, in the case of VOS, the region of interest appears many times throughout the sequence, unlike background motion, which only appears for a small number of frames. In another work [21], each segment of the sequence is represented by a set of binary masks, an appearance model in the form of a stripe, and a spatio-temporal transformation that captures the deformation of the appearance along the sequence. These three elements are then optimised for a set of segments in order to reconstruct the input video. The obtained masks can be viewed as object segmentation, taking into account both appearance and motion. Both methods require an optimisation on the input sequences at inference time, which takes time compared to the previously mentioned approach where the network is trained beforehand.

1.2 Motion Segmentation

In this section, we focus on motion segmentation, which is the task of partitioning 2D motion along a video, regardless of the source of each individual motion. It is then a "pure" multi-label segmentation problem based on the estimated image motion.

Motion segmentation has received considerable attention over several decades. Seminal works are based on clustering framework or Markov Random Fields (MRF) to perform image motion segmentation into layers [28], [29] as well as into connected regions [30], [31]. The advent of deep learning and the availability of efficient optical flow methods has recently led to new categories of methods. Most of these methods take as input the optical flow computed between two successive images and train neural networks to produce

a segmentation. In addition, some methods couple the input optical flow with image data to provide more reliable segmentation, while others exploit temporal relationships through trajectories or recurrent structures.

The motion segmentation task is related to the previously described Video Object Segmentation (VOS) task, but differs in that they do not focus on extracting a dominant "foreground object" that occupies a central place in the video. Thus, these techniques focus primarily on the motion information rather than the image or semantic characteristics of the frames that compose the sequence.

In addition, it is difficult to build a large labelled training database for motion segmentation, as it would require precise annotation of the different motions in the sequence, which is not trivial in some cases (e.g., articulated motion). Deep learning techniques focused on this task have to come up with new ideas to train a network in an unsupervised way, leading to a wide variety of approaches. In the DiVA paper [49], it is qualified as a good task to "explore the limits of fully unsupervised object detection".

Given the large body of work in this area, we will not attempt an exhaustive review. Instead, we focus on a handful of methods that we consider representative of the field and that have most influenced our research. We have constructed the taxonomy shown in table 1.1, which highlights the different approaches investigated in the litterature, and we will follow the structure of this taxonomy for this section.

We deliberately do not base the taxonomy on the obvious distinction between learning and optimisation-based approaches, as this would have the side-effect of separating past and recent methods. In contrast, we intend to highlight the methodological similarities or differences between all methods, and show how the seminal approaches inspired the current approaches to formulate motion segmentation.

1.2.1 Instantaneous motion segmentation

Our first focus is on methods that rely on the instantaneous motion between two successive frames of the video. Most of these methods belong to the category of motion decomposition as they produce a partition of this instantaneous motion into coherent segments. They use as input the pair of frames or a pre-computed optical flow field that represents the dense point-to-point correspondance field between the successive frames. In this category we distinguish three types of approaches.

Taxonomy			**Title**	OF	AF	Te	DL
Instantaneous Motion Segmentation	Parametric motion models (2D / 3D)	Explicit spatial prior (MRF)	[31]				
			[32]				
			[33]	✓			
		Prediction of region boundaries	[34]				
			[35]				
		Without explicit spatial prior	[36]	✓	✓		✓
			[4]	✓			✓
			[37]				✓
			[3]	✓	✓		✓
			[5]		✓		✓
			[29]	✓			
	Compensation of camera motion		[38]	✓			
			[39]	✓			
			[40]	✓			
			[41]	✓			
			[42]	✓			✓
			[43]				✓
			[44]				
	Non parametric approaches		[45]				
	Deep generative models		[46]	✓			✓
			[47]	✓			✓
			[48]	✓	✓		✓
			[49]	✓			✓
			[21]	✓	✓		✓
Trajectory Clustering			[50]	✓			✓
			[51]			✓	
			[52]			✓	
			[53]	✓	✓		✓
			[54]			✓	

Table 1.1 – Taxonomy of Motion Segmentation Methods. "OF" denotes methods using a precomputed optical flow field as input, "AF" denotes methods using pretrained appearance features, "Te" represents methods taking into account long temporal context, "DL" methods using deep learning models.

Use of 2D parametric motion models

In this first category, the methods represent the motion within each segment with a parametric motion model, and usually with a polynomial motion model that is related to the projection of the 3D motion in the viewed scene onto the 2D image [30].

Parametric Motion Models

Following the rigid-body dynamics, we can define the motion of any point P on the surface of a rigid body wit $P = (X, Y, Z)^T$ in the coordinate sytem attached to the camera and centered in 0, given the instantaneous translation vector $\vec{\mathcal{T}} = (U, V, W)^T$ and the rotation vector $\vec{\mathcal{R}} = (A, B, C)^T$ specifying the 3D rigid motion as :

$$\vec{V} = \vec{\mathcal{T}} + \vec{\mathcal{R}} \times \vec{OP} \tag{1.1}$$

Using this definition and a perspective projection model providing $x = f\frac{X}{Z}$ and $y = f\frac{Y}{Z}$, we can retrieve the 2D velocity vector $w(p) = (u(p), v(p))^T$ of the projection $p = (x, y)^T$, into the image plane, of the 3D point P as :

$$\begin{aligned} u(p) \triangleq \frac{dx}{dt} &= f\frac{U}{Z} - x\frac{W}{Z} - A\frac{xy}{f} + B(\frac{x^2}{f} + f) - Cy, \\ v(p) \triangleq \frac{dy}{dt} &= f\frac{V}{Z} - y\frac{W}{Z} - A(\frac{y^2}{f} + f) + B\frac{xy}{f} + Cx. \end{aligned} \tag{1.2}$$

Those equations link the 3D motion of a point to the 2D motion of its projection in the 2D image plane. It involves the 3D rigid motion between the object and the camera and the depth of the point. In practice, Z can be different for each point. The difference in the apparent motion of static objects in the foreground of the scene due to the camera motion and the difference in depth is called "motion parallax". Those equations inspired the use of parametric motion models describing the 2D motion in the image sequence. A specific 2D motion model is the 8-parameter quadratic model corresponding to the projection of a rigid motion of a planar surface :

$$\begin{aligned} u_\theta(p) &= \theta_1 + \theta_2 x + \theta_3 y + \theta_7 x^2 + \theta_8 xy, \\ v_\theta(p) &= \theta_4 + \theta_5 x + \theta_6 y + \theta_8 xy + \theta_7 y^2. \end{aligned} \tag{1.3}$$

Some recent methods [3], [55] use a full 12-parameter quadratic motion model where the two components of the motion model are independent. It can better fit complex motion and account for non planar (but still continuous) surface. The affine (6-parameter) motion model corresponding to a polynomial of degree 1 for both components is also often used.

The authors in [29] introduced a clustering approach that splits the motion between two images using affine motion models. They first estimate the optical flow field between frames using a local variational approach, and then build an iterative algorithm that alternates between estimating an affine motion model segment by segment and assigning pixels to one of the computed segments. This work inspired several recent deep learning efforts to train a network to perform layered motion segmentation based on parametric motion models.

A pioneering work [4] on the use of loss based on parametric motion models to train a neural network for segmentation was, ironically, not carried out in the objective of segmenting a video, but in the very active field of research of self-supervised feature learning for image segmentation. In fact, in this work, the authors relyed on the Gestalt principle of common destiny to advocate that the pixels moving in a coherent way are likely to belong to the same object. They have pre-trained the feature extractor of a static image segmentation module using unlabelled videos and a criterion that evaluates the segmentation based on the fitting error of an affine motion model to the flow of each given segment. They manage to show that the features obtained are a good pre-training step for segmentation. However, since they use a least-square computation for the motion model fitting, their method is sensitive to noise and outliers in the optical flow estimation, which affects the quality of the result obtained. It leads them to use a more basic loss based on the entropy of the histogram of the flow magnitude within each segment.

Several recent works have followed this lead. In [5], the authors train a network to segment an image from a video frame with a loss function based on the decomposition of the motion into piecewise constant motion models. They also learn to predict the optical flow field for each segment from a reconstruction loss. Later, in [3], the authors go further in this direction, using a quadratic motion model, a pre-trained optical flow from RAFT [56], and a pre-trained appearance feature extractor from DINO [6]. It greatly improves the segmentation results, but complicates the evaluation of the impact of self-monitoring to train feature extraction networks. The authors also presented a probabilistic approach to their work in [36], using affine motion models and testing on the MoVi [57] and KITTI [58] datasets.

Using the same criterion for learning, several methods directly segment the motion between two frames. Our motion segmentation method [55] that we will detail in Chapter 2, belongs to this category as it trains a network to segment optical flow fields using a loss function based on parametric motion models. In [37] the authors set up an unsupervised approach to segment motion into dominant and background motion using affine motion models from a pair of frames. Like [5], their network takes two consecutive frames as input and predicts both the motion segmentation and affine flow models for each layer based on image warping and a photometric loss function. The main differences with [5] are that their segmentation is based on both frames instead of just one of the frames and that they use affine motion models instead of constant piecewise models. This approach is interesting as the authors do not rely on precomputed optic flow for training, but they only provide results on binary segmentation of simple translational motion (moving car dataset). Therefore, it is hard to evaluate if the feature encoder is able to represent more complex appearance patterns.

Explicit spatial prior

Related methods combine the use of parametric models with prior conditioning of the output segmentation. In [30]–[32], the authors present methods for segmenting the motion between two successive images based on parametric motion models and a spatial constraint that forces neighbouring pixels to have the same labels using Markov Random Fields. The segmentation involves an iterative algorithm that alternates between estimating the parametric motion models and performing motion segmentation.

Markov Random Field and Bayesian Approach

In the Bayesian framework, the probability of a segmentation is decomposed into its likelihood (i.e., data fitting term) and prior terms. The data fitting term U quantifies how well the motion models attached to the segmentation x align with the observed input data y. The prior V_c assumes that x is a 2D Markov Random Field (MRF). Due to the equivalence between MRF and Gibbs distribution, the prior can be defined locally on cliques $c \in \mathcal{C}$ that consists of spatially inter-connected pixels. It allows for example to promote smoothness in the output segmentation. A common prior for segmentation is the Potts model that forces neighbouring sites to have the same label. A common criterion is the Maximum a posteriori (MAP), leading to the minimization of the negative log-likelihood of the segmentation probability, which is expressed using the Bayes rule as :

$$\begin{aligned} -\log p(x|y) &= -\log p(y|x) - \log p(x) + cst. \\ &= \sum_{i \in \Omega} U(x_i, y_i) + \sum_{c \in \mathcal{C}} V_c(x_c) + \text{cst}. \end{aligned} \tag{1.4}$$

Ω is the set of sites in the image grid. Authors then employ optimisation procedures (e.g., Simulated annealing, Iterative Conditional Modes, Graph-Cut) to minimise this negative log-likelihood and obtain a segmentation for a given input data y.

In [33] the authors introduce a richer prior with the idea of a hierarchical segmentation of motion, linking each segment to a parent motion, each child node has a parametric motion model that fits only the residual motion. This leads to a finer representation of motion, which can be useful in applications that link parts moving together. While these priors on segmentation have shown to be an asset in traditional approaches, it is still a challenge to apply these constraints in deep learning methods. Indeed, difficulty comes up with a differentiable version of the compactness constraints. To our knowledge, there are no methods that include them explicitly during the training procedure.

Another way to include a spatial constraint on the prediction is to predict the bound-

aries of the regions, thus forcing them to be enclosed and with a regular shape. In [34], the authors introduce a variational approach to segmenting instantaneous motion from a pair of frames by predicting the boundary between regions. Their framework combines a likelihood probability that evaluates the coherence of a segmentation using constant or affine motion models, and a prior that minimizes the length of the boundaries between motion segments. Their likelihood function is based on a criterion derived from the brightness constancy equation and the representation of motion in terms of parametric models, which is expressed as :

$$p(w|\nabla_3 I) \propto p(w) \prod_{p\in\Omega} p(\nabla_3 I(p)|w(p)) \propto e^{-\gamma\mathcal{L}(C)} \prod_{p\in\Omega} e^{-\cos^2(\alpha(p))}, \tag{1.5}$$

where $\Omega \subset \mathbb{R}^2$ denotes the image grid, $w(p)$ is the motion vector at point p, here in homogeneous coordinates, $\nabla_3 I(p)$ is a vector stacking the spatial and temporal intensity gradients at point p. The likelihood term includes $cos(\alpha(p))$, which is the angle between $\nabla_3 I$ and w at point p and represents the errors in the optical flow constraint at that point. The prior term includes $\mathcal{L}(C)$, the length of the boundary C between the computed regions weighted by parameter γ.

By representing the flow within each region using a parametric motion model, they further decompose the likelihood as :

$$\prod_{p\in\Omega} p(\nabla_3 I(p)|w(p)) = \prod_{i=1}^{n} \prod_{p\in\Omega_i} p(\nabla_3 I(p)|w_i), \tag{1.6}$$

where w_i is the motion model (affine or constant in this case) representing the motion within region i and Ω_i is the set of sites belonging to that region, and n the number of regions.

The authors represent the boundaries between regions using either splines or level sets, allowing them to minimize $\mathcal{L}(C)$ using gradient descent.

One of the advantages of this method is that it produces regular shaped regions, since it directly models the boundaries. Also, the fact that they combine motion estimation with segmentation alleviates the need to use pre-computed optical flow fields. Some limitations of the method are related to the use of the brightness constancy equation, which restricts the method to the segmentation of small motions and to images where the brightness of each point is constant over time. Other limitations are related to the initialization of the region boundaries, as it can affect results.

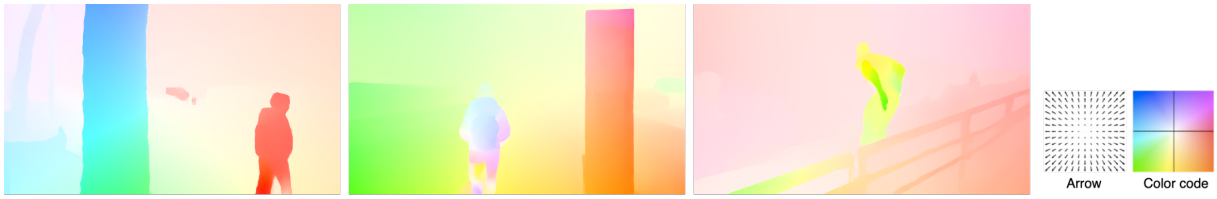


Figure 1.1 – Examples of parallax motion in natural scenes. Optical flow fields colored with the HSV color code [59] are shown on the bottom right. From left to right: two flows from our outdoor videos, one flow from a parkour sequence in Davis. In the left sequence, the parallax motion is the apparent motion of the tree trunk, in the center sequence the apparent motion of the pole, and in the right sequence, it is the motion of the fence.

In [35], the authors use a framework similar to [34] in that they rely on level sets and parametric motion models to represent the motion within each segment. In addition, they introduce a term that enforces that the motion boundaries coincide with spatial flow gradients.

Compensation of camera motion

Another perspective is to focus on the ability to detect independent moving objects in the scene. As we explained earlier, in the case of a mobile camera, the apparent motion of objects depends on their depth, so that static objects close to the camera can have an apparent motion that is different from the background motion. Consequently, these objects are segmented as individual moving objects using the methods above in subsection 1.2.1.

Several methods tried to address this problem and segment only objects that are really moving in the scene. A reference method in this field is [43], where the authors jointly trained four convolutional networks that take as input a set of images and independently provide estimates of the depth map of the scene, the camera motion between frames, the dense optical flow field, and the segmentation of the moving objects. Then, by combining the depth map and the estimated camera motion they compute the scene flow, representing the apparent motion of the whole static scene. The gap between the static scene flow and the dense optical flow should be high in the regions corresponding to independent moving objects. The four networks are trained using a process called "competitive collaboration" and give good results on the KITTI dataset [58]. However, in addition to the availability of the camera intrinsic parameters, this method requires an accurate estimation of the scene depth map and the camera pose at the same time, which can be difficult to obtain

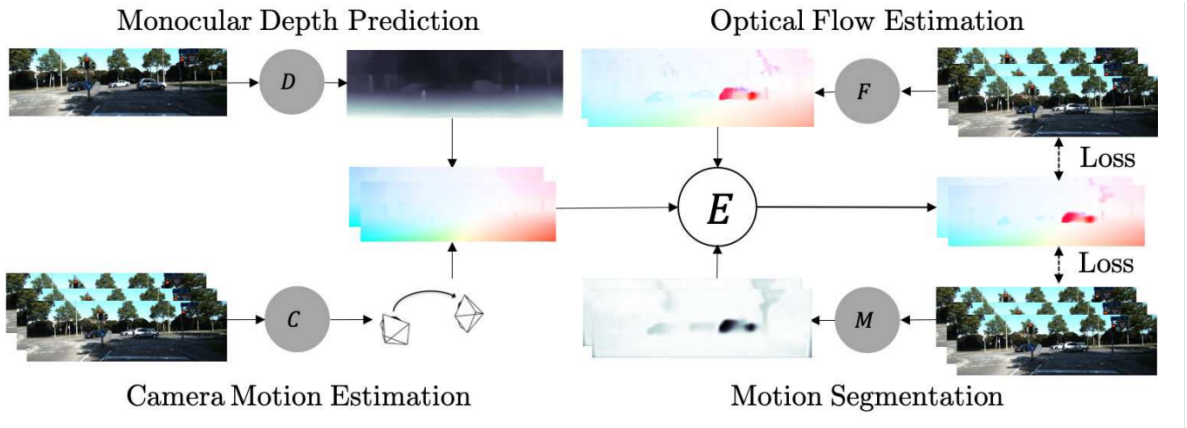


Figure 1.2 – Framework of the competitive collaboration process presented in [43]. Four networks taking either a frame or a sequence of frames as input are jointly trained. The "E" step represents the combination of the different modalities to form a reconstructed optical flow which is evaluated on the input sequence using photometric loss. Reproduced from [43].

[60]. To our knowledge, there is no application of this methods in natural scenes shot with a handheld camera such as in the DAVIS2016 dataset. It makes it uncertain whether this method could adapt to different intrinsic camera parameters and more complicated camera movements than translation. Furthermore, there are many hyperparameters in this method and no simple approach to find them.

Compensation of the camera rotation Another interesting approach is presented in [38]–[42]. The aim is to ignore parallax motion without having to estimate the full depth map. To do this, they compute the 3D rotational motion of the camera using a robust estimation method. Once the camera rotation estimated, they show that the angular field of the translational motion is independent of the depth of the object, and then, they can segment this flow field without having to worry about parallax motion.

Compensate Rotational Motion

We can decompose the 2D flow field $\vec{w}$ as the sum of a translational flow field $\vec{w}_t$ and a rotational flow field $\vec{w}_r$ such that :

$$\vec{w} = \underbrace{\frac{1}{Z}\begin{bmatrix} fU - xW \\ fV - yW \end{bmatrix}}_{\vec{w}_t} + \underbrace{\begin{bmatrix} -A\frac{xy}{f} + B(\frac{x^2}{f} + f) - Cy \\ -A(\frac{y^2}{f} + f) + B\frac{xy}{f} + Cx \end{bmatrix}}_{\vec{w}_r} \tag{1.7}$$

Thus, we can observe that the orientation κ_t of the translational field is independent of Z :

$$\kappa_t = \arctan(\frac{fV - yW}{fU - xW}). \tag{1.8}$$

Several works (e.g., [38], [39]) compensate the rotational component of the flow and segment the orientation field of the resulting translational flow to get a segmentation not affected by parallax motion.

They investigated this perspective first using classical probabilistic models for segmentation and a robust approach for camera rotation estimation [38], [39]. Then, they introduced neural networks into the method. A pre-trained network allows them to segment the rotation-compensated flow, then, they train a network to perform segmentation using supervised training on synthesised sequences with different motions [40]. Finally in [42], they combine rotation estimation with a supervised convolutional network for segmentation and show results on real data. This line of research shows the usefulness of estimating camera motion to improve segmentation, although they still need to have the intrinsic parameters of the camera for the projection model, to apply this technique in practice. Furthermore, it seems intuitive that rotation compensation is a critical step for unsupervised segmentation, as it is hard to find objective criteria allowing to distinguish motion parallax from independent motion. However, in the context of supervised learning, where independent motions are clearly annotated in the ground truth, one could wonder why the network would need a rotation-compensated flow field, since it could learn to achieve the rotation compensation internally.

3D-consistency It is also possible to detect moving objects in 3D space using a rigidity constraint based on projective geometry. In [61], the authors present several approaches for detecting moving objects based on plane and parallax decomposition. The novelty is that they derive a geometric relationship between pairs of points in several images, which makes it possible to check whether these points are moving like rigid objects, even if they are at different depths, thus allowing to detect areas that are moving independently without being affected by parallax motion. In another approach [62], the authors first decompose the motion into segments and then use rigidity constraints and compatibility tests based on the motion characterization of each segment to distinguish foreground from background motion. While these approaches provide valuable insights into the detection of independently moving objects, they have certain limitations. Like previous works that compensate for camera rotation, they rely on a good estimate of camera extrinsics, which can be difficult to compute in cases where there are many moving objects. It also requires knowledge of the camera intrinsic parameters.

Deep generative motion model

A drawback of the parametric approach is the limited representativeness of these models. We have presented techniques above that address this issue by representing motion associated with variation in the depth field, but other types of motion, such as articulated motion, remain a challenge. In MoSeg [46], the authors argue that this lack of flexibility of classical motion models affects the performance of segmentation methods based on them. In this context, they build a layered segmentation method similar to the one presented above, in the sense that they try to reconstruct the input flow using a layered representation, but with the crucial difference that the flow specific to each layer is freely generated by a deep learning decoder, instead of being induced by a set of motion model parameters. They focus on the binary segmentation of the optical flow, which consists in labelling the pixels as foreground moving objects or background, and demonstrate good results on classical VOS datasets.

However, this class of region-based approaches faces a structural drawback: the trade-off between model expressiveness and segmentation quality. Indeed, the more expressive the generative model, the less relevant the segmentation. In the case of an extremely rich generative model, the entire input stream could be represented in a single class, making segmentation unnecessary and thus defeating the training strategy. Conversely, an overly simple model can only model the flow within small segments, leading to very fragmented

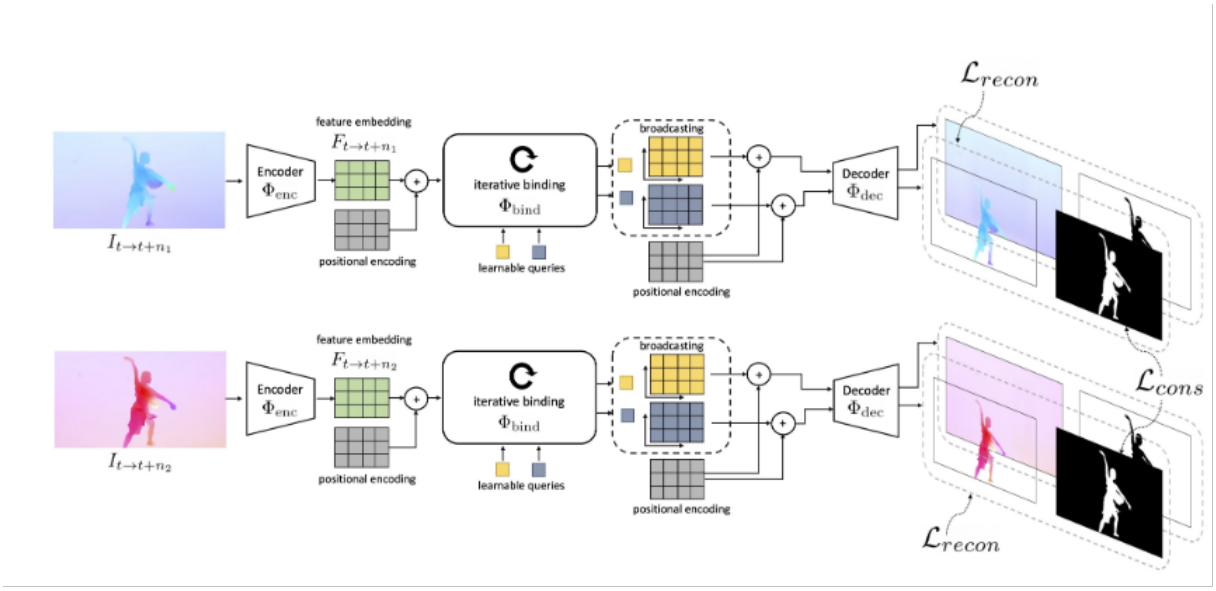


Figure 1.3 – Framework of MoSeg [46]. Slot attention module is applied on the feature extracted from the input flow field. Extracted slots are then given as input to a decoder network to produce the flow reconstruction and the segmentation masks. Reproduced from [46]

segmentation.

An alternative to this problem has been proposed under the name of Contextual Information Separation (CIS) [47]. The authors build an adversarial network where the generative network tries to predict the optical flow inside the mask using only information outside it. This way, even with a rich class of generative model, it is impossible to achieve it if the movements inside and outside the mask are completely unrelated.

Contextual Information Separation

In [47], the authors make a distinction between region-based segmentation criteria used in the classical literature, where one tries to reconstruct the original flow using decomposition and a parametric model for each segment [63], [64]. The aim of their approach is the prediction of the flow field inside each region from the flow field outside it. The authors introduce two mathematical formulations as follows:

Classical region-based segmentation

$$\int_{\Omega^f} |w^{in}(p) - \phi(p, w^{in})|^2 dp + \int_{\Omega^b} |w^{out}(p) - \phi(p, w^{out})|^2 dp, \quad (1.9)$$

Contextual information separation

$$\int_{\Omega^f} |w^{in}(p) - \phi(p, w^{out})|^2 dp + \int_{\Omega^b} |w^{out}(p) - \phi(p, w^{in})|^2 dp. \quad (1.10)$$

Here, Ω^f represents the foreground region, w^{in} represents the flow within that region (Ω^b and w^{out} for the background, respectively). The generative model is ϕ, which can take various form. Within the region-based framework, ϕ is a constant value of each segment in [63], it is a smooth function in the Mumford-Shah functional [64], a parametric motion model in [3], [36], [55] and the output of a generative network in [46]. Within the contextual information separation framework, it is a convolutional network in [47] and a generative network in [49].

Let us note that a limitation of this approach is the difficulty to hide part of the optical flow. This is because a probabilistic segmentation mask is involved during training. The CIS method showed good results on binary segmentation of the optical flow field on VOS benchmarks. It was then extended first to the DyStab [48] method, which in addition uses appearance for segmentation, and the DiVA method [49], dealing with multiple motion segmentation and relying on the slot attention architecture introduced in [46].

Another recent work [21] explores a hybrid approach, combining a very simple (piecewise constant) parametric model with an estimation of the residual flow for each segment,

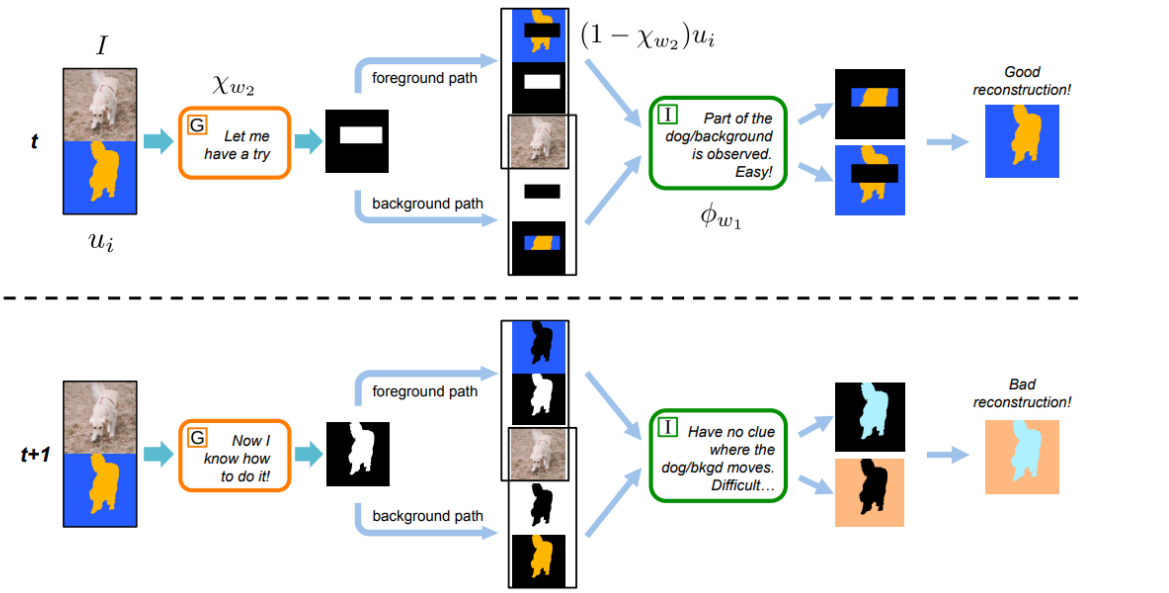


Figure 1.4 – Framework of CIS [47] with two cases of execution. Reproduced from [47].

which represents the reconstruction error between the parametric and the dense optical flow. In this model, they do not control the complexity of the residual flow, but it is just restrict its values to be within $[-\lambda, \lambda]$, thus avoiding the residual flow to directly output the target optical flow field. However, the setting of λ is critical and depends on the input data. The authors do not provide a way to easily tune this parameter. Their approach additionally relies on image appearance information with an appearance refinement stage. In addition, the network is trained to predict a cleaner segmentation based on a Conditional Random Field (CRF) and a semantic constraint loss using pre-trained appearance features.

Non-parametric approaches

Non-parametric approaches to motion segmentation base their segmentation on criteria that are not described by parametric motion models. In [45], the authors construct a functional that evaluates the smoothness of the motion in each slice, as measured by the optical flow gradients within the slice. The functional is minimized using an EM algorithm to compute the segmentation along with the flow field while satisfying the smoothness constraint. Green's functions are used to make the computation of the maximization steps efficient.

1.2.2 Clustering trajectories

Several works achieve motion segmentation in an image sequence by analysing trajectories. The method presented in [51] is representative of the classical approach to this problem. In this paper, the authors compute trajectories along the video sequence and build an affinity metric function that evaluates the distance between two trajectories. Then, they apply a clustering algorithm on the pairwise affinity matrix to group trajectories together, leading to a sparse motion segmentation. Finally, they adopt a variational approach that exploits image gradients to convert the sparse labels into a dense segmentation.

This method underlines the three key elements of the trajectory clustering methods :

— A representation of trajectories along the sequence and an affinity metric between trajectories,

— A clustering algorithm to group trajectories together,

— A method to go from the sparse labeling to a dense segmentation.

Many methods have been developed that vary these key elements. For example, the authors in [52] use a similar distance trajectory representation and distance metric as in [51], but they also include colour information. Then, for clustering, they follow a minimum-cost multicut formulation, which allows the number of clusters to be set automatically. It is also possible to build a method closer to the parametric approaches described above by replacing the clustering algorithm with the estimation of a global background motion along the sequence. By comparing to the background motion, one can identify points that differ from this global motion as independently moving objects, as done in [54]. The background motion model along the sequence is defined by a set of homographies $H_k, k \in \{1 \cdots T\}$, where H_i is the homography representing the global motion between I_i and I_{i+1}. In this method, they exploit a CRF to obtain a dense segmentation from the sparse labeling. One advantage of this approach is that it is more robust than classical clustering. However it assumes that the background occupies most of the image, and, because of the use of homographies to represent global motion, it can only accommodate the motion of a plan in 3D space.

Recently, another approach [53] revisited the trajectory clustering methodology using deep learning, focusing in particular the transition from sparse to dense labeling. The authors cluster the trajectories, then, train a U-net to provide a dense segmentation from a frame of the video using the sparse annotation as ground-truth labels. This method allows to improve the dense segmentation results by relying on the convolutional network ability

to learn an appearance representation from an image and to regularise a segmentation by filling the gap between sparse labels. They also introduce a Siamese GRU network to compute the distance between trajectories. The disadvantage is that the network is trained independently on each sequence, which makes the method very slow (30 min/sequence on a GPU). They show that both the GRU and the U-Net improve the segmentation results, demonstrating the benefit of training the modules of this approach. Another method [50] sets up a dense network to learn the trajectory representation in the embedding space, where the motion segmentation problem is solved at inference by applying K-means to the output embeddings. By training a loss that predicts the distance between two trajectories, in contrast to directly predicting the segmentation map, they open the door to predicting a variable number of clusters during the K-means step. However, since the ground truth does not involve hierarchical information, one cannot directly expect the network to learn by itself to produce a coherent motion partition with a variable number of segments.

1.2.3 Comparison between learning-based and optimisation-based approaches

As we mentioned at the beginning of this section, an important distinction is the difference between learning-based and optimisation-based approaches. We call "learning based" methods that train neural networks to perform the task and then perform inference only at test time. The fact that learning based methods do not require iterative computation at test time makes them really fast and efficient for practical applications, but also removes the need for initialisation, which is often a hindrance for classical approaches. The ability to implicitly learn shape priors or appearance conditioning from the data is another advantage of learning-based methods. Empirical results show that, even without a loss term that explicitly introduces this regularisation, a neural network performing segmentation is more likely to produce compact and regular shaped output at inference time, if it has been trained on this type of target.

On the other hand, learning-based approaches have drawbacks, the most commonly cited is that it is long and computationally expensive to train. Another issue is that the results at inference can be hard to explain and quantifying the uncertainty of the network prediction is still an open question. More specifically to our application, it can also be more difficult to integrate priors on segmentation, as one needs to find a loss function that is differentiable and works with the evolution of the segmentation masks

along the training. An important drawback of learning-based methods is the difficulty of generalising to data outside the training distribution. In practice, a trained network is often unable to correctly process inputs that are too different from the training data, or to deal with challenges that are not present in the training set. This has a direct impact on the unsupervised image segmentation methods presented above, as they often struggle to adapt to unseen objects, relying on pre-trained image features or test time adaptation to achieve good results on the test set.

1.3 Motion Saliency

Motion saliency (MS) aims to highlight local motions departing from their surrounding context, thus prone to reveal a significant event. MS has many applications in computer vision. It can be useful in the navigation of autonomous vehicles to anticipate moving obstacles, or for public safety to trigger alert in abnormal situations. It can facilitate the subsequent analysis of videos where motion may play the key discriminative role.

MS estimation can be formulated as a meta-task that can encompass different tasks.

1.3.1 Video saliency

Video saliency is by definition object-oriented and therefore mixes appearance and motion features.

Previous approaches exploited the fact that salient regions have different low-level appearance and motion features (also known as cues) from neighbouring regions. The authors of [12] first divide the image into regions using hierarchical segmentation, and then rely on descriptive histograms for each region by combining motion and appearance cues. Regions with a large difference from their neighbours are labelled as salient. The pipeline of this method is presented in Fig. 1.5. In [65], the authors exploit a criterion to measure the coherence of regions using the entropy of the histogram of gradients (HoG) of the luminance channel and the entropy of the histogram of the flow magnitude. In this work, a region with coherent motion, low spatial luminance gradient entropy and greater motion than its neighbours is termed salient.

Other works introduce graphs with distance measures based on appearance and motion indices to perform segmentation. In [66], the authors decompose the video into temporal superpixels and then construct a graph using metrics based on appearance and motion

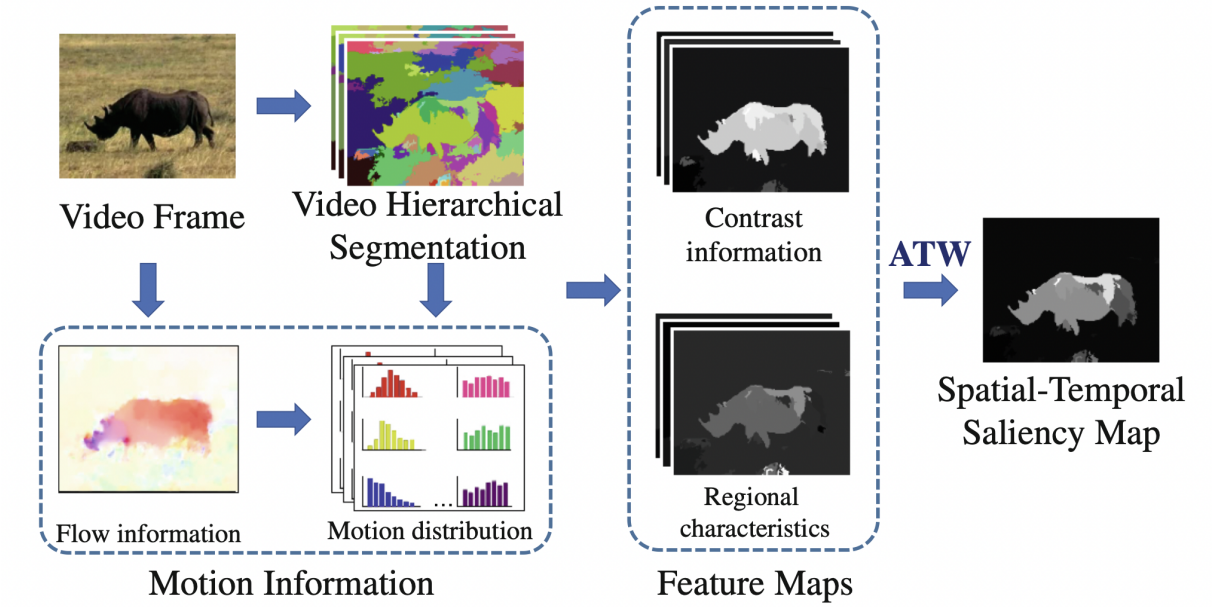


Figure 1.5 – Pipeline of the dynamic saliency method. Reproduced from [12].

cues. Then, they apply a spectral foreground detection method to this graph in order to identify the salient region. In [67], the authors first identify the initial salient region under the assumption that the background region is connected to the image boundary. Then, they use a graph to diffuse saliency labels based on pre-calculated image edges, an inter-frame term based on optical flow and a term based on visual similarity between superpixels.

In [68], the authors assume that salient objects have different low-level appearance and motion features (orientation and magnitude for flow vectors) than the background. They use a Tukey-inspired measure to detect outlier pixels in the images and label them as belonging to moving objects. They also introduced methods based on pre-trained semantic features that improve segmentation.

The methods mentioned above rely on the extraction of meaningful motion features and on assumptions inherent to the data set. For example, they rely on the fact that motion saliency is often correlated with appearance saliency, or that the salient object often has a central position in the image. These assumptions are true for the VOS dataset, but could be not verified for other types of data. For example, the MoCA [22] dataset presents sequences where the labeled objects exhibit salient motion but have an appearance similar to the background.

In [69], the authors also rely on appearance and motion cues for motion saliency. In this paper, they formulate the problem as anomaly detection and train a 3D convolutional network on image sequences. The network is trained in a self-supervised manner using pretext tasks with the goal of developing discriminative anomaly-specific features. The first two tasks deal with the consistency of the temporal evolution of the video. The first one consists in guessing if the sequence is played forward or backward, and the second one is to detect if there is an irregularity in the motion of the sequence. The third task is to evaluate if the network is able to reconstruct the central frame of the sequence. Finally, a knowledge distillation task based on pre-trained image features allows them to include appearance in the training process. At inference time, the salient frames are defined as those where the network fails to perform these tasks. Several methods related to the used of deep learning for anomaly detection are described in [70].

1.3.2 Eye-tracking based saliency

Another way to measure saliency is to use eye-tracking data. By measuring the positions where a human gazes while watching a video, one can retrieve a region of interest that is likely to contain the salient object. This metric was originally used for evaluation, but with the advent of deep learning, several works introduced methods to train neural networks to predict these gaze patterns and thus localize salient regions. We will have a look at some of these approaches here, a more complete overview is available in [71].

In [72], the authors introduce a large eye tracking database with videos that present a variety of content. Using this database, they train a deep learning approach combining a convolutional network and a recurrent network (convLSTM) to learn to predict saliency maps from input images in a supervised manner using eye-tracking data as ground-truth labels. In [11], the authors also leverage eye-tracking data for training, but they introduce a two-stream network that relies on 3D convolutions to detect salient areas on a sequence of images. By exploiting this two-stream architecture, they have a lighter network that requires less video gaze data to train. In addition, they incorporate a positional prior to the convolutional network for segmentation, pushing the output to be concentrated around the center of the image using Convolutional Gaussian Priors layers. Inspired by previous work on saliency prediction, they use a combination of three different saliency metrics including the ground truth density map and the ground truth binary fixation map for the training loss.

Other works show that it is advantageous to train the gaze prediction network and

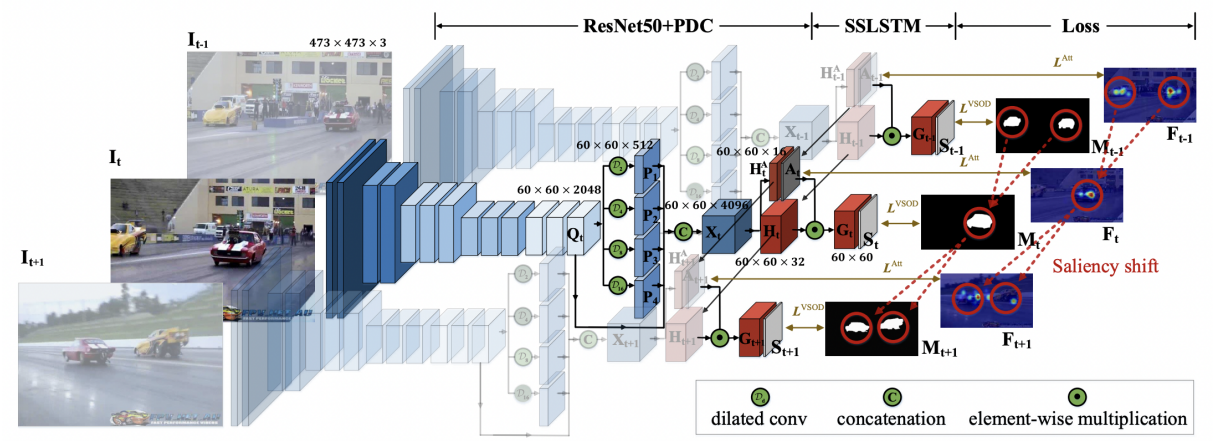


Figure 1.6 – Overall framework of [73], the image are first processed using a convolutional network with pyramid dilated convolution, then with a LSTM network. On the right, the segmentation and human eye-tracking annotation corresponding to the input images are presented. The figure highlights the presence of saliency shifts where the salient object changes along the sequence. Reproduced from [73].

perform segmentation at the same time. In [73], the authors present a large dataset of videos annotated with both instance segmentations and eye tracking. They train the model combininb convolutional and recurrent network in a supervised manner on both modalities, trying to predict both segmentation and human attention. The overall framework of this method is presented in Fig. 1.6. In [74], both modalities are also predicted using an alternative architecture exploiting attention in the segmentation phase.

1.3.3 Pure motion saliency and Trajectory-based saliency

By pure motion saliency, we mean that motion is the discriminative feature, and consequently, only motion information is exploited as input of the saliency detection method. This issue may be also referred to as anomalous motion detection. In that category, we can mention the use of social forces computed from the flow field [75], and of the linear approximation of the dynamical system defined by the optical flow [76].

A classical type of motion saliency to detect is salient motion resulting from a distinctive motion within a coherently moving set. The set can typically be a crowd of people, a herd of animals, a flock of birds, vehicle traffic, or a set of cells. In [77], the authors use local affine motion histograms that characterize the motion around a point at different scales, to detect regions that have different motion than their neighbors, and show that

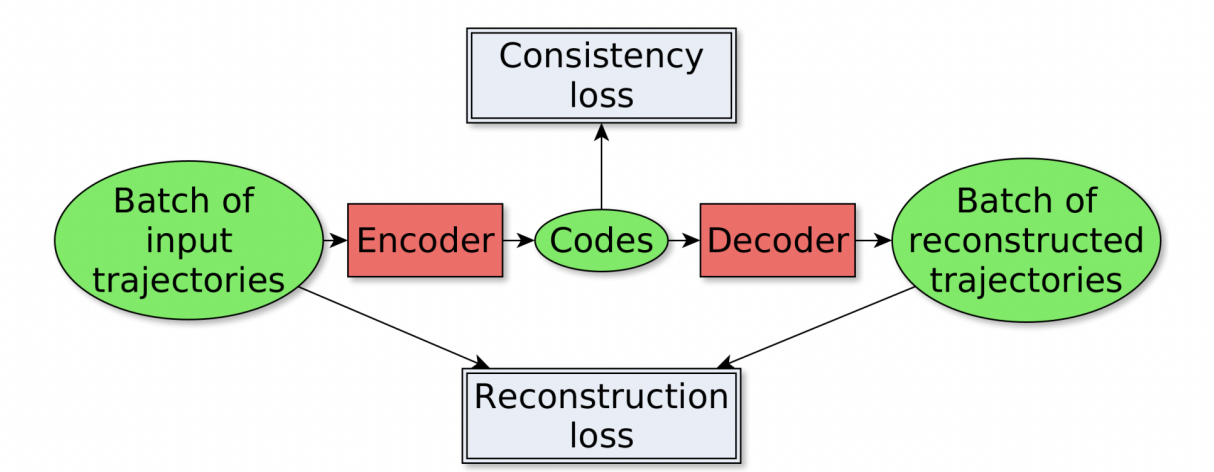


Figure 1.7 – Training scheme of the method presenting the processing steps of batch of trajectories along with the two associated losses. Reproduced from [81].

they can use it to detect anomalous motion in a crowd. We refer the reader to [78] for a more complete description of the methods related to this task.

Some methods use the similarity of trajectories along the sequence to group points and detect salient motion. In [79], the authors compute a coherence metric between trajectories over the sequence based on motion similarity. They group points with high similarity. They use it to segment trajectories of points in 3D and groups in crowds. In [80], groups in crowd scenes are detected using an iterative process based on Markov chains to characterize the motion in each group. A set of features is then used to describe the relative motion of each group.

In [81], the authors train a salient trajectory detection method using an auto-encoder with a consistency loss that ensures that the latent codes for non-salient trajectories are similar. They achieve this by adding a regularization term that penalizes the latent codes that are far from the median latent code computed on the batch. This facilitates the use of the latent code for later classification, since the salient trajectories should have a very different representation from the non-salient ones. At inference time, they use a statistical criterion based on the distance to the median latent code to detect salient trajectories in an unsupervised setting. They apply this work to pedestrian trajectories in a train station and show that they can successfully extract salient trajectories. The training scheme of this method is presented in Fig. 1.7.

1.4 Partial Conclusion

In this chapter, we have introduced work on motion analysis with papers on motion segmentation, video object segmentation, and motion saliency. Although each of these tasks presents its own set of methods, they share similarities in that they all process image sequences. Across the set of presented works, we can identify three key components.

First, the choice of input data. Methods such as [17], [24], [31], [37], [43], [45] work directly on image pairs to extract moving objects. Others [16], [19], [22], [38], [39], [46], [47], use a precomputed flow field. Methods such as [3], [21] take only images as input to extract salient objects from appearance cues. They use motion as a signal to guide the process. Each choice of input data presents its own challenges. Working from image pairs requires extracting similarities between images. Appearance-based methods depend on the extraction of relevant image features. Optical flow field-based methods must deal with errors in motion estimation.

Second, the category of approach. We highlighted the importance of classical approaches in all tasks [27], [29], [33], [51], [68], [77], which provided a solid foundation for motion analysis. With the development of deep learning and the availability of large annotated datasets [15], [22], several supervised learning methods have been proposed for VOS and motion saliency such as [16], [17], [19], [20], [22], [23]. We also presented more recent unsupervised learning techniques [3], [46], [47] designed to deal with tasks where there is no available ground truth. Unsupervised approaches share similar challenges with classical approaches, as they require a good modeling of the problem to design the loss function.

Third, how methods model temporal dimension. We have presented methods that consider only instantaneous motion between two frames, others that consider a sequence of frames, and finally methods [51]–[53], [81] that consider long time spans with sparse spatial representation (e.g., trajectories).This point has important implications for several aspects of the methods. For classical methods, it has an impact on the modeling assumptions, and for deep learning methods, it has an impact on the loss function and the choice of network architectures.

Despite these advances, several challenges remain. First, the current unsupervised learning approach still lags behind the supervised approach for VOS tasks. Learning to group dependent motions (e.g. articulated motions or swarn motions) in a fully unsupervised manner is an interesting challenge, and could reduce the gap between the two

categories of approaches on VOS. Second, we can see that recent image-based approaches [3], [21] struggle to generalize to unseen image distributions. Thus, they often rely on the use of externally pre-trained features or on fine-tuning the method on the test data (i.e., test-time adaptation). Adressing the generalization challenge could improve the applicability of these methods to different input data modalities. Finally, a still open problem is the integration of a rich temporal dimension. Most of the methods we have presented process only a few frames or use trajectories, but incorporating a longer context could make it possible to address complicated motion analysis. Developing network architectures that effectively account for long-term context and designing losses that account for temporal coherence are necessary steps to address this challenge.

In our work, we focus on the first and third challenges as they are the most relevant to motion analysis. We have introduced methods to improve unsupervised motion segmentation and to integrate the temporal dimension into our networks (Chapters 2, 3, and 4). We have attempted to tackle unsupervised motion saliency estimation using task-specific interpretation information from a classification network (Chapters 5 and 6). In addition, we have developed a DL-based optical flow method dedicated to small moving objects in large images (Appendix 7.4). This appendix contains also a brief survey of recent optical flow methods based on deep learning, along with a detailed presentation of the RAFT method [56] and its architecture.

Chapter 2

EM-driven Unsupervised Learning for Motion Segmentation

In this chapter, we present a CNN-based fully unsupervised method for motion segmentation from optical flow. We assume that the input optical flow can be represented as a piecewise set of parametric motion models, typically, affine or quadratic motion models. The core idea of our work is to leverage the Expectation-Maximization (EM) framework in order to design in a well-founded manner a loss function and a training procedure of our motion segmentation neural network that does not require either ground-truth or manual annotation. However, in contrast to the classical iterative EM, once the network is trained, we can provide a segmentation for any unseen optical flow field in a single inference step and without estimating any motion models. We will investigate different loss functions including robust ones and propose a novel efficient data augmentation technique on the optical flow field, applicable to any network taking optical flow as input. In addition, our method is able by design to segment multiple motions. We have tested our motion segmentation network on four benchmarks, DAVIS2016, SegTrackV2, FBMS59, and MoCA.

2.1 Problem Statement

Motion segmentation is among the main computer vision tasks. Its goal is to divide a frame into motion-related coherent segments. Motion coherence must be understood with respect to a given property expressed by motion features, parametric motion models, or even higher-level motion information. Depending on the formulation of the problem or the need of the application, segments can be layers, i.e., non necessarily connected subsets of points, or regions, i.e., connected segments, forming a partition of the image grid. Motion segmentation is a relevant step for many applications covering video interpretation, biomedical imaging, robot vision, and autonomous navigation, to name a few.

Motion segmentation is a complex problem that has been investigated for decades as described in Chapter 1, but there are still open questions. Indeed, it combines topology and information aspects in an intricate way. By topology, we mean the partition of the frame constituting the output of the segmentation. By information, the type of features or motion models it relies on. This all results in a chicken-and-egg problem: estimating easily and correctly the involved motion models requires an available partition, getting an accurate and reliable partition implies available motion models driving the segmentation.

As we want to address general-purpose image motion segmentation without anticipating any given application, we cope with optical flow segmentation (OFS). Indeed, the optical flow carries all the information related to motion between two successive frames of a video. Segmenting optical flow enables specific computer vision goals, as for instance, segmenting independent moving objects in the scene. Of course, the image motion depends on one hand on the relative motion between scene objects and the camera, and on the other hand on the object depth. As a consequence, any significant difference in depth results in a distinctive image motion. A static object in the foreground of the scene produces a specific flow pattern compared to the static background, and its image motion is supposed to form a separate segment. As done in Chapter 2, we call this situation "motion parallax".

We assume that the input optical flow can be represented as a piecewise set of parametric motion models, typically affine or quadratic ones, each of them characterizing motion in one segment. Thus, we formulate OFS as a piecewise linear regression problem, where finding supports (segments) and estimating the motion models are intertwined issues.

This problem can be addressed with latent variables, which usually imposes an alternate optimisation strategy. The Expectation-Maximization (EM) algorithm is certainly the flagship solution for a statistical approach of this problem [82]. Several extensions to the original EM were proposed as the Classification EM (CEM) introduced in [83], where emphasis is put on the clustering issue of the problem beyond the mixture model one. However, classical EM relies on handcrafted features, and leads to time-consuming iterative algorithms. On the other hand, deep learning, and more precisely, convolutional neural networks (CNN), have now become the most effective key solution for image and motion segmentation [43], [84]–[86]. Nevertheless, the training step remains an important issue. Supervised learning provides high accuracy, but manual annotation of optical-flow segmentation maps as ground truth is very cumbersome, and almost unreachable at a large scale. Unsupervised training is thus preferable but more challenging to optimize, in

particular to formulate the appropriate loss leading to the intended outcomes. Moreover, an unsupervised method is certainly the best way to deal with videos, and specifically optical flows, unseen during the training phase, thus ensuring better generalisation.

In this chapter, we aim to bring the two, EM and CNN, together in order to design a principled and efficient unsupervised motion segmentation method. By unsupervised, we mean that we do not resort to any ground-truth and manual annotation, both for the training stage in the loss function, for the selection of the optimal trained network model and for the choice of network hyperparameters. On one hand, the parametric motion models will carry the coherence for each motion segment. However, the key point is to confine their estimation to the training stage. On the other hand, we take EM as the well-founded basis for the design of the loss function and consequently the training stage of the motion segmentation network. In the related framework of mixture density networks [87], a neural network is combined with a mixture density model. However, it does not rely on EM. Once trained, our network segments each flow of the video without any iteration and any motion model estimation.

Thus, the main purpose of our EM-driven network is to uncover motion coherence. Taken alone, it can segment motion within videos. It can also be incorporated in a larger framework to solve any video segmentation or understanding problem that could benefit from a motion coherence cue. This could be achieved either by using our network as a module inside a bigger pipeline, for motion saliency detection for instance, or by using our proposed loss as a regularization term for the training of a neural network.

We will evaluate and compare our method to existing methods on four datasets: DAVIS2016 [15], FBMS59 [51], SegTrackV2 [88] and MoCA [22].

The chapter is organized as follows. We present in Section 2.3 how we leverage the EM algorithm to design the network loss function, its architecture and the training procedure. Section 4.4 reports extensive experiments with comparison to other existing methods on four benchmarks. In Section 2.5, we discuss several issues related to the main components of our approach including a comparison with the choices made by other methods.

2.2 Motion Segmentation as an EM Problem

The core idea of our work is to leverage the Expectation-Maximisation framework to design in a well-founded manner the loss function and the training procedure of our OFS neural network. In this section, we first describe one way to use the EM algorithm for

optical flow segmentation.

Optical flow $f \in \mathbb{R}^{2\times W\times H}$ is a vector field defined over an image grid Ω of size $I = W \times H$. We denote by $f_i \in \mathbb{R}^2$ the motion vector associated to each site $i \in \Omega$ of this grid. We make the assumption that any optical flow field can be decomposed in a set of K segments or layers, each one grouping a (possibly non connected) part of the image grid and exhibiting a coherent motion. In order to enforce coherence, we choose to represent the motion field within each segment k with a parametric model defined by parameters θ_k. We denote $\theta = \{\theta_k, k = 1, \ldots, K\}$. In practice, we use polynomial motion models, typically affine (first-degree polynomial) or quadratic (second-degree polynomial) models. Their interest lies in both an easy physical interpretation and an efficient estimation. For instance, a specific 8-parameter quadratic motion model corresponds to the projection, into the image plane, of the rigid motion of a 3D planar surface. Thus, we can account for any slanted almost planar surface that exhibits a smooth depth variation, not only fronto-parallel ones. By almost planar, we mean a negligible depth variation of the object surface compared to its distance to the camera. Our method could nevertheless accommodate other types of parametric models.

We now consider the likelihood of the optical flow field f given the set of parameters θ, denoted by $p(f|\theta)$. In order to make explicit the partition of f into k segments and the associated individual θ_k's, we introduce latent variables z_i such that $p(z_i = k|f_i, \theta_k)$ represents the probability that site i belongs to layer k. Assuming conditional independence, the logarithm of the likelihood can be written as below in step 1 of eq.(2.1). Then, we introduce the z_i variables (steps 2 and 3 of eq.(2.1)), and we finally straightforwardly make any positive distribution q appear (step 4 of eq.(2.1)). We have:

$$\begin{aligned}
\log(p(f|\theta)) &= \log \prod_i p(f_i|\theta) \\
&= \log \prod_i \sum_k p(f_i, z_i^k|\theta_k) \\
&= \sum_i \log \sum_k p(f_i, z_i^k|\theta_k) \\
&= \sum_i \log \sum_k q(z_i^k) \frac{p(f_i, z_i^k|\theta_k)}{q(z_i^k)}, \qquad (2.1)
\end{aligned}$$

where $z_i^k \triangleq [z_i = k]$. Maximizing $\log(p(f|\theta))$ w.r.t. θ is obviously complicated, even if it boils down to k maximizations w.r.t. the θ_k's. Indeed, variables z_i are hidden. To maximize eq.(2.1) w.r.t. θ_k, variables z_i must be available. Accordingly, we need to

maximize also w.r.t. the z_i's.

However, we can use the Jensen's inequality ($h(\mathbb{E}[x]) \geq \mathbb{E}[h(x)]$ for any concave function h), as done in classical EM [89], in order to build a lower bound $ll(\theta)$ of the log-likelihood $\log(p(f|\theta))$. We get $\log(p(f|\theta)) \geq ll(\theta)$ with:

$$\begin{aligned} ll(\theta) &= \sum_i \sum_k q(z_i^k) \log \frac{p(f_i, z_i^k|\theta_k)}{q(z_i^k)} \\ &= \sum_i \sum_k q(z_i^k) \log p(f_i, z_i^k|\theta_k) \\ &- \sum_i \sum_k q(z_i^k) \log q(z_i^k), \end{aligned} \tag{2.2}$$

where the first term of eq.(2.2) is the expectation over $q(z_i)$ of $\log p(f_i, z_i|\theta)$ and the second term is the entropy that we note $\mathcal{H}$. The resulting expression of the lower bound is:

$$ll(\theta) = \sum_i \mathbb{E}_{q(z_i)}[\log p(f_i, z_i|\theta)] + \sum_i \mathcal{H}(q(z_i)). \tag{2.3}$$

In the classical EM algorithm, one usually takes $q(z_i^k) \triangleq p(z_i^k|f_i, \theta_k)$, i.e., the posterior distribution. Then, one alternates between an expectation step where $q(z_i^k), \forall i, k$, is estimated, and a maximization step where $ll(\theta)$ is maximized w.r.t. the θ_k's. This procedure monotonically increases the log-likelihood until it reaches a local optimum [89].

2.3 CNN-based Motion Segmentation

In our case, we adopt a neural network model $g_\phi(f)$, taking as input the optical flow f and parameterized by ϕ, to produce the image motion segmentation that is our primary goal. This network has a softmax activation in order to output a valid probability distribution over layers for each site.

The motivation is that, by doing so, we can access a large family of distributions. Most importantly, after the training stage, our network is able to infer the motion segmentation without iterating and without involving motion models,

in contrast to the classical EM algorithm. In addition, it is much faster. At the training stage, we deal with two sets of parameters: the parameters of the motion models θ and the parameters of the network model ϕ. At the inference stage, we are only concerned by the network parameters ϕ. The overall flowchart of our method, including training and

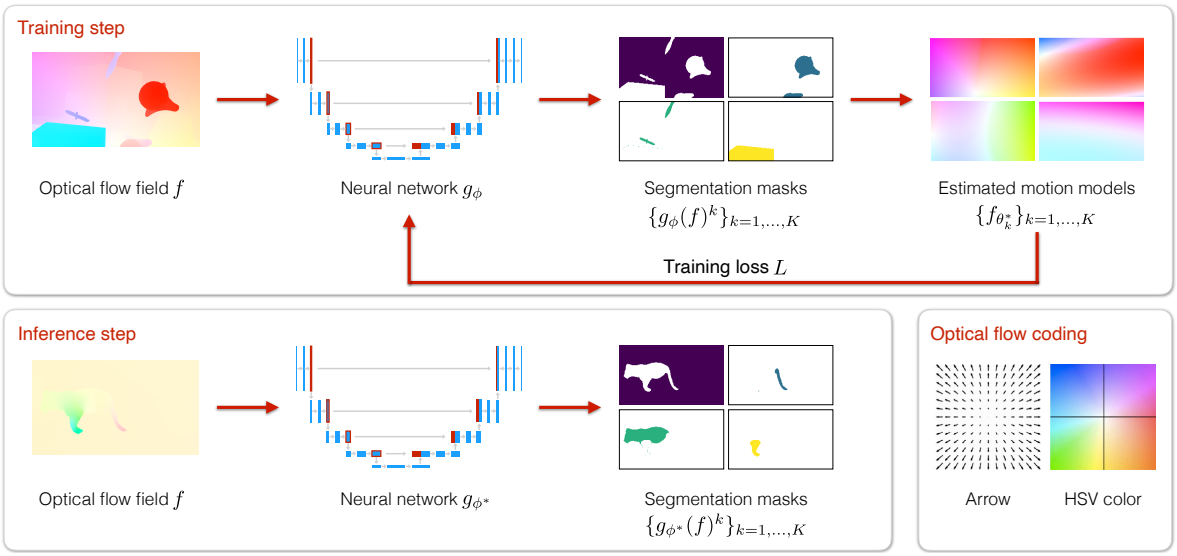


Figure 2.1 – Flowchart of the proposed CNN method for the training (top) and inference (bottom) steps. *Training step*: First, we segment the optical flow field f with the neural network g_ϕ. Then, we get the optimal parametric motion models $\{f_{\theta_k^*}\}_{k=1,\dots,K}$ within each probabilistic segmentation masks $\{g_\phi(f)^k\}_{k=1,\dots,K}$ using (2.16). Finally, we update the parameters ϕ of the neural network using (2.17), where the loss function is defined in (2.12). This training step is performed iteratively over each batch $\mathcal{B}$ (of size 1 in this illustration). *Inference step*: We directly apply the trained network g_{ϕ^*} to any new unseen optical flow field f to obtain the probabilistic segmentation masks $\{g_{\phi^*}(f)^k\}_{k=1,\dots,K}$. There is no estimation of the motion models $\{f_{\theta_k^*}\}_{k=1,\dots,K}$ in the inference step in contrast to the training step. For the sake of visualization, optical flows and polynomial motion models are represented with the HSV color code, but actually, the flow field f used as input of the neural network is taken as a 2D vector field. We have a two-channel input. *Optical flow coding:* correspondence between the arrow visualization of the optical flow field and the HSV color map.

inference stages, is given in Fig.2.1.

2.3.1 EM-driven network specification

Coming back to eq.(2.2) and following the choice expressed above, we take $g_\phi(f)_i^k$ as $q(z_i^k)$, where $g_\phi(f)_i^k$ is the probability (prediction) given by the network for site i to belong to segment k given the input optical flow f. The lower bound now depends on two sets

of parameters, θ and ϕ, and writes:

$$\begin{aligned} ll(\theta, \phi) &= \sum_i \sum_k g_\phi(f)_i^k (\log p(f_i, z_i^k|\theta_k) - \log g_\phi(f)_i^k) \\ &= \sum_i \mathbb{E}_{g_\phi(f)_i}[\log p(f_i, z_i|\theta)] + \sum_i \mathcal{H}(g_\phi(f)_i). \end{aligned} \tag{2.4}$$

We alternatively optimize $ll(\theta, \phi)$ with respect to θ and ϕ for the training stage as follows:

$$\theta^* = \arg\max_\theta \sum_i \sum_k g_\phi(f)_i^k \log(p(f_i, z_i^k|\theta_k)) \tag{2.5}$$

$$\begin{aligned} \phi^* = \arg\max_\phi &\sum_i \sum_k g_\phi(f)_i^k \log(p(f_i, z_i^k|\theta_k^*)) \\ &+ \sum_i \mathcal{H}(g_\phi(f)_i). \end{aligned} \tag{2.6}$$

As previously described in [90], the entropy of the predicted segmentation at each site i, $\mathcal{H}(g_\phi(f)_i)$, naturally arises in eq.(2.4), and then, in eq.(2.6). Entropy measures statistical uncertainty and is maximised for $g_\phi(f)_i^k = \frac{1}{K}, \forall i, k$. It acts as a regularization term balancing the likelihood term to avoid falling too quickly into (inappropriate) local optima.

Regarding the optimisation on θ, we can reach a local optimum using an off-the-shelf iterative algorithm. However, we can only perform a gradient descent step for the optimisation with respect to the network weights ϕ.

In order to gain intuition on how the network is learning to produce the motion segmentation, following [89], we rewrite the lower bound as:

$$\begin{aligned} ll(\theta, \phi) &= \sum_i \sum_k g_\phi(f)_i^k \log \frac{p(f_i, z_i^k|\theta_k)}{g_\phi(f)_i^k} \\ &= \sum_i \sum_k g_\phi(f)_i^k \log \frac{p(z_i^k|f_i, \theta_k)}{g_\phi(f)_i^k} \\ &+ \sum_i \log p(f_i|\theta) \sum_k g_\phi(f)_i^k \\ &= -\sum_i \mathbb{KL}[g_\phi(f)_i || p(z_i|f_i, \theta)] + \log(p(f|\theta)). \end{aligned} \tag{2.7}$$

Consequently, the optimisation step over the network weights is defined by:

$$\phi^* = \arg\min_\phi \sum_i \mathbb{KL}[g_\phi(f)_i || p(z_i|f_i, \theta^*)], \tag{2.8}$$

where we minimize the KL-divergence between the segmentation produced by the network and the segmentation linked to the optimal parameters θ^*. Thus, the network is trained to produce a segmentation for a given set of parameters. As the quality of the network segmentation improves, so does the quality of the estimated θ^*, pushing the network weights to produce better segmentation in turn.

2.3.2 Flow likelihood and loss function

In the previous section, we described the overall training process. In this section, we address the definition of the different terms of the loss function.

First, we decompose the joint probability in eq.(2.4) into a likelihood and a prior:

$$p(f_i, z_i^k|\theta_k) = p(f_i|z_i^k, \theta_k)p(z_i^k). \tag{2.9}$$

The likelihood $p(f_i|z_i^k, \theta_k)$ assesses how the estimated parametric motion model in a given region fits the observed flow in this region. In this work, we use a uniform prior for $p(z_i^k)$. Nevertheless, we could adopt a more complex prior, if we wanted to influence the size of each region for instance.

An important point of our design is to specify the form of the likelihood $p(f_i|z_i^k, \theta_k)$ that is used to compare the input optical flow with the parametric flow for a given set of parameters θ. In practice, since our parametric motion models are dependent on the position of the points on the 2D space, we introduce a deterministic function $c(i)$ that maps the site i to a polynomial expansion involving its coordinates. More specifically, for a 6-parameter affine motion model, we have $c(i) = \Big[1, x_i, y_i\Big]$; for a full 12-parameter quadratic model, we have $c(i) = \Big[1, x_i, y_i, x_i^2, x_i y_i, y_i^2\Big]$. The likelihood evaluates the distance between the input flow vectors f and the parametric flow vectors $f_{\theta_{k_i}} \triangleq \theta_k^T \cdot c(i), \forall k, i$. Its general form is given by:

$$p(f_i|z_i^k, \theta_k) = \frac{1}{Z}\exp(-\frac{1}{\alpha}\delta(f_i, \theta_k^T \cdot c(i))), \tag{2.10}$$

where $\delta : \mathbb{R}^{2*2} \to \mathbb{R}$ is a distance function to define, and α is a free parameter related to the uncertainty in the flow measure and the resulting adequacy of the parametric motion model. If δ is a translationally invariant function, which we verified for all tested distance functions, then Z is only dependent on the function δ and hyperparameter α and not on the input. The proof can be found in Appendix 2.4.3. This allows us to perform optimisation without explicitly computing Z.

The choice of the distance function δ is central in our approach, as it is used both for the estimation of the parametric motion models and for the training of the network (see eq.(2.5) and eq.(2.6)). Robust loss functions can be beneficial as thoroughly investigated in [91]. We consider the following distance functions:

— Squared L_2 : $\delta(f_i, \theta_k^T \cdot c(i)) = ||f_i - \theta_k^T \cdot c(i)||_2^2$
— L_2 norm : $\delta(f_i, \theta_k^T \cdot c(i)) = ||f_i - \theta_k^T \cdot c(i)||_2$
— L_1 norm : $\delta(f_i, \theta_k^T \cdot c(i)) = ||f_i - \theta_k^T \cdot c(i)||_1$

L_1 (due to the absolute function involved) and L_2 (due to the square root of the sum involved) norms bring robustness to outliers in the flow field, in contrast to the squared L_2.

We define the loss of our model as:

$$L(f, \theta, \phi) = -ll(\theta, \phi), \tag{2.11}$$

where $ll(\theta, \phi)$ is given by eq.(2.4). Taking into account eq.(2.10), we can formulate the loss function as:

$$\begin{aligned} L(f, \theta, \phi) = \frac{1}{\alpha} \sum_i \sum_k g_\phi(f)_i^k \delta(f_i, \theta_k^T \cdot c(i)) \\ + \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k + I \ \log(K \ Z), \end{aligned} \tag{2.12}$$

where Z is the normalization term in eq.(2.10) and α allows us to balance the likelihood, prior and entropy parts of the loss. We use $\alpha = 10^{-2}$ in all our experiments, but in practice the network model is fairly robust to the choice of this hyperparameter.

Herefater, we give more details on the derivation of eq.(2.12).

$$
\begin{aligned}
L(f,\theta,\phi) &= -ll(\theta,\phi) \\
&= -\sum_i \sum_k g_\phi(f)_i^k \log p(f_i, z_i^k|\theta_k) + \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k \text{ using (4)} \\
&= -\sum_i \sum_k g_\phi(f)_i^k \log p(f_i|z_i^k,\theta_k) - \sum_i \sum_k g_\phi(f)_i^k \log p(z_i^k) \text{ using (9)} \\
&\quad + \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k \\
&= \log Z \sum_i \sum_k g_\phi(f)_i^k + \frac{1}{\alpha} \sum_i \sum_k g_\phi(f)_i^k \delta(f_i, \theta_k^T \cdot c(i)) \text{ using (10)} \\
&\quad - \sum_i \sum_k g_\phi(f)_i^k \log p(z_i^k) + \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k.
\end{aligned} \tag{2.13}
$$

As aforementioned, we use a uniform prior for $p(z_i^k)$, i.e., $p(z_i^k) = \frac{1}{K}$. We can thus rewrite the loss L as:

$$
\begin{aligned}
L(f,\theta,\phi) =& \log(K\ Z) \sum_i \sum_k g_\phi(f)_i^k + \frac{1}{\alpha} \sum_i \sum_k g_\phi(f)_i^k \delta(f_i, \theta_k^T \cdot c(i)) \\
&+ \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k.
\end{aligned} \tag{2.14}
$$

As $\sum_k g_\phi(f)_i^k = 1,\ \forall i$, we have $\sum_i \sum_k g_\phi(f)_i^k = I$, where I is the number of sites i in the frame. We finally obtain:

$$
\begin{aligned}
L(f,\theta,\phi) = I\ \log(K\ Z) &+ \frac{1}{\alpha} \sum_i \sum_k g_\phi(f)_i^k \delta(f_i, \theta_k^T \cdot c(i)) \\
&+ \sum_i \sum_k g_\phi(f)_i^k \log g_\phi(f)_i^k.
\end{aligned} \tag{2.15}
$$

We describe in Appendix 7.1 an alternative approach to end up with the same loss function, based on the variational inference framework involving the Evidence Lower Bound (ELBO).

2.3.3 Network training

During the training, we minimize the loss function L over a dataset of optical flow fields. For each input flow field f of the training dataset, we minimize $L(f,\theta,\phi)$ with respect to each parameter. This alternate optimisation is performed over every batch $\mathcal{B}$

as follows:

$$\theta^* = \arg\min_{\theta} \sum_{f \in \mathcal{B}} L(f, \theta, \phi^t) \tag{2.16}$$

$$\phi^{t+1} = \phi^t - \gamma \nabla_\phi \sum_{f \in \mathcal{B}} L(f, \theta^*, \phi), \tag{2.17}$$

where t is the iteration number and γ the learning rate.

In practice, we use an optimizer to get θ^* and an automatic differentiation to compute the gradients with respect to ϕ. As described in subsection 2.3.1 and in our computation graph presented in Fig.2.2, we consider θ^* as fixed in the gradient step with respect to ϕ, making $\nabla_\phi L(f, \theta^*, \phi)$ trivial to compute using automatic differentiation. Practical details are provided in subsection 2.4.1.

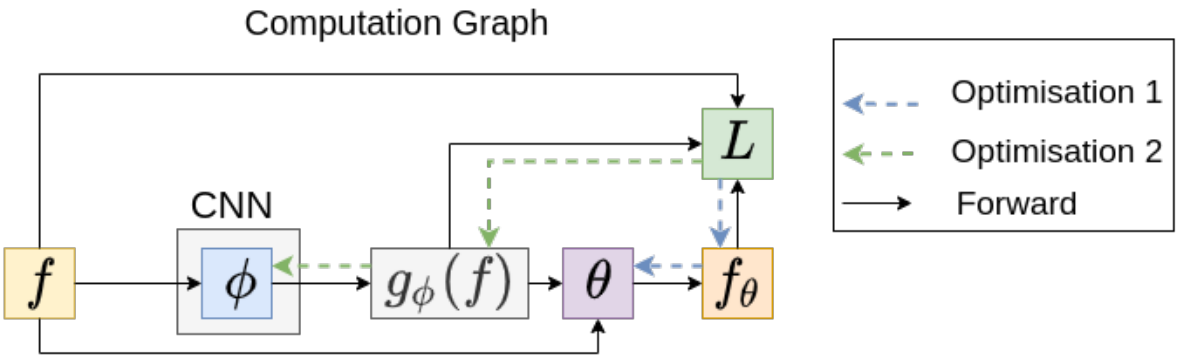


Figure 2.2 – Illustration of the computation graph for the training of our network. $m \triangleq g_\phi(f)$ denotes the set of arrays (as many as masks) collecting the probability for each site of the input flow field to belong to each mask. θ is the set of the motion model parameters, and ϕ the set of the network parameters. In our method, we are alternatively optimising w.r.t. θ (optimisation 1) and ϕ (optimisation 2).

2.3.4 Data augmentation

As proven beneficial in many computer vision problems, we proceed to data augmentation to train the motion segmentation network. However, the input data are not images but optical flows in our case, which led us to define an original data augmentation procedure. The goal is to make the network as invariant as possible to the global motion field, since we identify it as an important clue for generalisation.

Therefore, we add to each optical flow field of the dataset a parametric motion model whose parameters are drawn at random. For the sake of consistency, we take quadratic motion models. This mimics a large variety of camera movements. This procedure has the advantage of multiplying the flow configurations, while keeping the same flow structure as in the initial sample. In other words, the network is trained with a diversity of flows, while having the same target segmentation to predict. In Appendix 7.1, we give a formal proof that our loss function is invariant to the added parametric global motion. As shown in Section 2.4.3, this data augmentation scheme contributes to improve the overall performance of the network. Let us emphasize that it could also be used to train any network taking optical flow as input with the benefit of such invariance.

2.3.5 Related work on EM and deep learning

To conclude this section, we explain how our approach differs from other works that somehow make use of EM within a neural network framework. In [92], the EM paradigm is involved in the network designed for semantic segmentation. However, the purpose is quite different since they exploit the EM algorithm in the attention mechanism. It allows them to iteratively estimate a compact set of bases used to compute the attention maps.

The EM framework plays also a role in [93], [94], and just recently in [95], in the design of the neural network architecture concerned with perceptual grouping tasks. However, those approaches differ in several ways. First, they remain iterative. One starts from parameters specifying each latent space component (as our motion parameters θ), but the network iteratively refines them in the inference stage, so that this network remains dependent on the initialization. Another big difference with our method lies in the network architecture itself, which includes generative branches to produce an approximation of the input data from the latent space components and an iterative branch (recurrent network in [93], [94] and sampling in [95]) to update the latent space components themselves. Since those methods rely on an iterative inference process implementing EM, they rather belong to the class of algorithm unfolding techniques [96].

2.4 Experimental Results

2.4.1 Implementation details

Optical flow fields are computed on the original video frames using the RAFT method [56]. A detailed description of the RAFT method and its architecture can be found in Appendix 7.4. More specifically, we use the RAFT implementation [1] with network weights fine-tuned on the MPI Sintel dataset [97]. At the time we started this work, the RAFT method was among the top methods on the MPI Sintel benchmark [2], whose code was available. For efficiency, we downsample the computed optical flow fields to obtain 128×224 vector fields provided as input to the network, as also done in the MoSeg method [46]. The resulting segmentation is subsequently upsampled to the original frame size for evaluation w.r.t. the ground truth. It allows us to perform much more efficient training and inference stages. In all experiments, we use the L_1 norm loss function, unless otherwise specified.

One may wonder about the impact of the chosen optical flow method on the motion segmentation results. Obviously, the network output is likely to depend on its input, but to what extent? Early experiments carried out at the beginning of this work showed that the motion segmentation results were barely downgraded when using the PWC-Net method [98]. A more challenging issue is to use a non-supervised optical flow method as the ARFlow method [99] that is necessarily far less accurate than supervised optical flow methods as RAFT and PWC-Net. We used the official PyTorch implementation of ARFlow [3] with the network weights provided in the file "checkpoints/Sintel/pwclite-ar.tar". In that case, the performance of our motion segmentation method clearly decreases, but interestingly, it is less affected than the MoSeg method as shown in Table 2.1.

We take the full quadratic motion model with 12 parameters to represent the optical flow within each segment k:

$$\begin{aligned} f_{\theta_k}(x,y) = (&\theta_{k_1} + \theta_{k_2}x + \theta_{k_3}y + \theta_{k_4}x^2 + \theta_{k_5}xy + \theta_{k_6}y^2, \\ &\theta_{k_7} + \theta_{k_8}x + \theta_{k_9}y + \theta_{k_{10}}x^2 + \theta_{k_{11}}xy + \theta_{k_{12}}y^2)^T, \end{aligned} \tag{2.18}$$

where point (x, y) belongs to segment k. We take this parametric motion model since it

1. https://github.com/princeton-vl/RAFT
2. http://sintel.is.tue.mpg.de/
3. https://github.com/lliuz/ARFlow

	Ours	MoSeg
RAFT	69.3	68.3
ARFlow	55.4	53.2
ARFlow*	57.9	-

Table 2.1 – Impact of the optical flow method on the accuracy of the resulting motion segmentation, and comparison with the behaviour of the MoSeg method whose scores are drawn from [46]. Experiments were carried out on the DAVIS2016 dataset and the Jaccard index (intersection over union) is used for the evaluation score. *In this experiment, ARFlow is only used at inference.

can better fit complex motion. Indeed, it is likely to better encompass the background motion when the camera motion includes both translation and rotation with a static background involving objects at slightly different depths, and for articulated motions as well.

Our method is fully unsupervised. We do not resort to any manual annotation for training nor for model selection. Indeed, in all experiments, we selected the stopping epoch from the loss function evaluated always on the same validation set: the official training split of DAVIS2016 [15].

We consider generalisation to unseen datasets as an essential property of any motion segmentation method. To apply this idea, we trained once and for all our model on a single dataset, namely Flying Things 3D (FT3D) [100], for all experiments. In addition, the use of a synthetic dataset alleviates the problem of choosing the relevant real training data for a given experiment. For completeness, we also performed training experiments on real datasets, and obtained similar or slightly better performance depending on the chosen real training set.

Let us recall that the optimisation on θ does not occur at inference stage. The network prediction for each site of the flow field to belong to each segment k is directly used to yield the motion segmentation map. We simply select for each site segment $\hat{k}$ with the highest probability. No postprocessing is performed on the resulting segmentation. As shown later, the obtained segments are generally smooth, certainly due to the implicit regularization capacity of the network.

We take as input the optical flow in its vector field representation $f \in \mathbb{R}^{W \times H \times 2}$. Thus, we have a two-channel input for the network. Our loss function and our training procedure could be adapted to any neural network designed for segmentation. We choose

the well-known convolutional architecture U-net [86] for g_ϕ. We use a slightly modified implementation of the one available under PyTorch Lightning [101]. We take seven down-sampling layers and start with a feature depth of 64. The selection of the structure is again unsupervised using the validation loss on FT3D dataset. As did in [55], we use InstanceNorm between convolutional blocks in order to tackle variations in optical flow magnitude over the dataset.

We use Adam [102] optimizer with a learning rate of 10^{-4} to train the network. The optimisation on θ is done with Pytorch implementation of L-BGFS [103]. Batches comprise flow fields randomly sampled from the training dataset.

Our network [4] is time efficient, being a simple convolutional network, with an average computation time of $0.008s$ per 128×224 input flow field on a Tesla-V100, if we consider a batch of size 32. Without any parallelisation (batch size of 1), it can run at 36fps making it usable for real-time applications. In particular, it is faster than the fastest method introduced in [46], because we do not use an iterative attention module, thus reducing our computational complexity. In contrast to methods involving self attention, there is a linear relationship between the complexity of our network and the size of the optical flow field, which allows us to readily process input of large dimensions.

2.4.2 Comparative evaluation

We want to objectively evaluate the performance of our method for segmenting optical flow fields. Due to the lack of benchmarks dedicated to OFS, we compare our method on four VOS datasets described below.

DAVIS2016 [5] [15] includes 50 videos (3455 frames) split in 30 train and 20 validation videos with diverse objects. Only the principal moving object is annotated in the ground truth. We use the official criteria for evaluation on this dataset: the Jaccard score and the contour accuracy score.

SegTrackV2 [6] [88] and **FBMS59** [51] respectively comprise 14 videos (1066 annotated frames) and 59 videos (720 annotated frames), each involving one or several moving objects. For FBMS59 we use the 29 sequences of the validation set for evaluation. In cases where there are several moving objects, we group them into a single foreground mask for evaluation, as done in [46].

4. https://github.com/Etienne-Meunier/EM-Flow-Segmentation
5. https://davischallenge.org/index.html
6. https://paperswithcode.com/dataset/segtrack-v2-1

Method	Training	Input	DAVIS2016 $\mathcal{J}$	DAVIS2016 $\mathcal{F}$	SegTrack V2 $\mathcal{J}$	FBMS59 $\mathcal{J}$	MoCA $\mathcal{J}$
Ours	Fully Unsupervised	Flow	69.3	70.7	55.5	57.8	61.8
MoSeg [46]			68.3	66.1	58.6	53.1	63.4
FTS [104]			55.8		47.8	47.7	-
TIS_0 [68]			56.2	45.6	-	-	-
TIS_s [68]		Image & Flow	62.6	59.6	-	-	-
CIS - No Post [47]			59.2		45.6	36.8	49.4
CIS - With Post [47]			71.5		62.0	63.6	54.1
DyStab - Dyn [48]	Supervised Features	Flow	62.4		40.0	49.1	-
DyStab - Stat&Dyn [48]		Image & Flow	80.0		73.2	74.2	-
ARP [27]		Image & Flow	76.2	70.6	57.2	59.8	-
MATNet [19]	Supervised	Flow	82.4	80.7			64.2
COSNet [24]		Image	80.5	79.5	-	75.6	50.7

Table 2.2 – Results on DAVIS2016 validation dataset, SegTrackV2, FBMS59 validation dataset and MoCA for several unsupervised and supervised methods (scores taken from [68], [47], [46] and [48]). $\mathcal{J}$ is the Jaccard index (region similarity) and $\mathcal{F}$ accounts for contour accuracy. The higher the value, the better the performance. For further explanation on the evaluation metrics, we refer the reader to the DAVIS2016 website. Following the evaluation protocols, reported scores are the average of scores over all samples in the corresponding dataset, except for DAVIS2016 where it is the average of each sequence average score. Our network is trained once and for all on the synthetic dataset FT3D whatever the benchmark.

Moving Camouflaged Animals (MoCA) [22] presents camouflaged animals in natural scenes. For a fair comparison, we use the subset released by [46] with 88 videos and 4803 frames. Ground-truth bounding boxes are provided instead of masks for evaluation. Accordingly, we convert our output to a bounding box around the largest connected region of our output mask as done in [46].

Apart from a few videos in SegTrackV2 and FBMS59, those datasets focus on one primary moving object. Indeed, videos depict one single independently moving object in the foreground. Consequently, the ground truth comprises only two segments: foreground primary moving object versus background. To be coherent with this status, we apply our method with two masks, i.e., $K = 2$. To choose the foreground mask, we rely on a simple heuristic where we designate the biggest mask as the background one.

We compare our method with several other supervised and unsupervised methods: MoSeg [46], TIS (two versions) [68], CIS [47], FTS [104], DyStab [48], ARP [27], MATNet [19] and COSNet [24]. All these methods were described in Chapter 1 (Related Work).

Results are collected in Table 2.2. For a fair comparison, we underline several points in this table. First, we differentiate methods trained on ground-truth segmentation masks such as COSNet [24] or MATnet [19], and unsupervised methods. We also indicate methods that use features trained in a supervised way, as it provides a strong advantage, and then do not fit in an unsupervised scenario. Indeed, DyStab [48] resorts to supervised classification features to initialise its networks, and ARP [27] requires a motion boundary detection algorithm previously trained in a supervised way. Secondly, as we evaluate here motion segmentation based on optical flow, we dissociate methods that take RGB images as input from methods using only optical flow like ours. Some works like [48], [68] propose versions with (TIS_s, DyStab-Stat&Dyn) and without (TIS_0, DyStab-Dyn) RGB frames taken as input, illustrating the influence of this input modality on the final results. As in [46], we distinguish the CIS version involving a strong CRF-based postprocessing, "CIS-With Post", and one without postprocessing, "CIS-No Post", since postprocessing drastically increases runtime making the method unusable for practical applications. It is measured in [46] that CIS [47] has a runtime of 11s/frame with postprocessing against 0.1s without.

Table 2.2 shows that our method outperforms all comparable methods on DAVIS2016 and FBMS59. Our method is the second best for the two other datasets. It is even close to the best one and far ahead the other comparable methods for MoCA. By comparable methods, we mean unsupervised methods without unusable postprocessing. Let us stress that we train our method on an external dataset, unlike CIS that is trained on test data as well. Scores obtained by our method for every sequence of the four datasets are collected in Appendix 7.1.

In Fig.2.3, we report visual results to figure out how our method behaves on different typical examples from the VOS datasets. We can observe four examples of failure cases with respect to the DAVIS2016 ground truth (for *scooter-black, kite-surf, blackswan* and *parkour* videos), although the extra parts segmented by our method make sense w.r.t. OFS task. Let us recall that the VOS task takes into account by construction only the primary moving object and not all moving objects in the scene. In *scooter-black*, the car segmented in the background is moving; in *parkour*, the fence is segmented as it exhibits an important motion parallax; in *kite-surf* and *blackswan*, the ripples on the water are segmented too. This type of complex examples will be more appropriately handled with the multiple motion segmentation described in subsection 2.4.5.

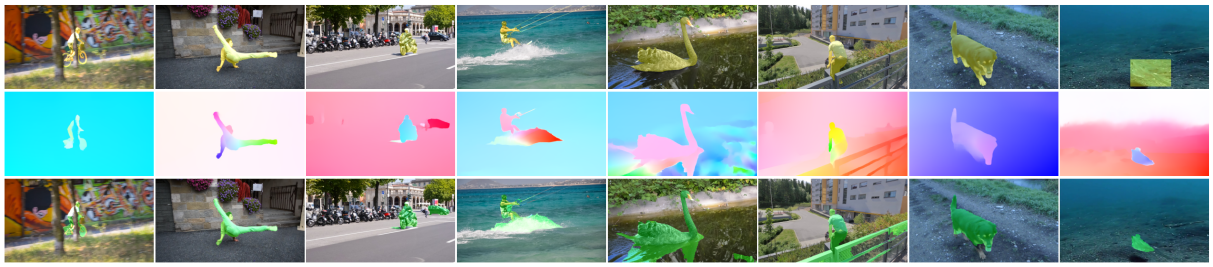


Figure 2.3 – Examples of motion segmentation results obtained with our method with two masks, on the videos *bmx-trees, breakdance-flare, scooter-black, kite-surf, blackswan, parkour, dogs02 and cuttlefish1*, from the DAVIS2016, FBMS59 and MoCA datasets. First row: a frame of the video with the ground-truth superimposed in yellow. Second row: the input flow field displayed with the HSV color code [59] that is depicted in Fig.2.1. Third row: the segmentation produced by our method superimposed in green on the corresponding image.

2.4.3 Ablation study

In order to identify the contribution of the different components of our method, we performed an ablation study. We investigated the following four changes:

— DA: removal of the data augmentation on the optical flow described in subsection 2.3.4.
— Quad. Model: replacing the full quadratic motion model with the affine one.
— L_1 norm (a): substitution of the robust loss function (L_1 norm) by its non-robust counterpart, the squared L_2 given in subsection 2.3.2.
— L_1 norm (b): substitution of the robust loss function (L_1 norm) by the L_2 norm given in subsection 2.3.2.

We performed each change on our full method one by one.

As shown in Table 2.3, the quadratic motion model plays a pivotal role in the performance of the network. The other components still play an important role in an equal measure. We can observe that the L_2 norm provides better performance than the squared L_2. As mentioned before, we train the network on the FT3D dataset where estimated optical flow fields exhibit low noise, which mitigates the impact of a robust loss for training. In previous experiments, the performance gap between a model trained with L_1 loss and squared L_2 loss on a real dataset was even more significant.

without	DA	Quad. Model	L_1 norm (a)	L_1 norm (b)	full method
$\mathcal{J}$ Mean ↑	58.4	53.1	58.8	60.6	61.1

Table 2.3 – Ablation study for different components of our method. Each time, we suppress or modify only one component, respectively, removal of the data augmentation, substitution of the quadratic motion model by the affine motion model, substitution (a) of the L_1 norm by the squared L_2, substitution (b) of the L_1 norm by the L_2 norm. $\mathcal{J}$ is given as the average of the average frame Jaccard index obtained for each of the four datasets presented in Section 2.4.2.

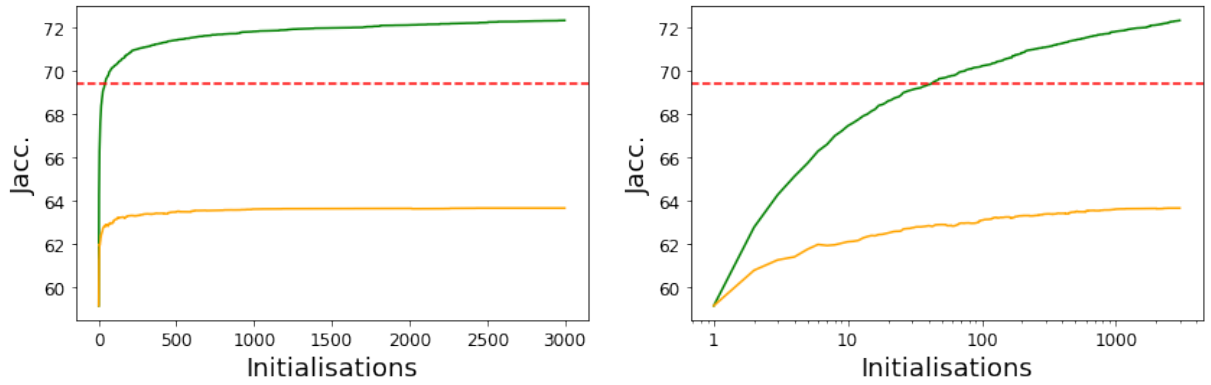


Figure 2.4 – Sensitivity with respect to the initialization of the classical EM algorithm in contrast to our trained model. First, we plot the evolution of the (speculative) optimal Jaccard index of the classical EM algorithm of Section 3 of the main text (green curve) with respect to the number of available random initialisations of the parametric motion parameters θ on the DAVIS2016 validation set. For all videos, the optimal (speculative) Jaccard index is obtained by selecting the best segmentation for each input flow among all available initialisations. Secondly, the yellow curve plots the evolution of the Jaccard index of the classical EM algorithm over the available initialisations, when selecting the initialisation with the highest likelihood, the only means available in practice without ground-truth. The dashed red horizontal line corresponds to the value of the Jaccard index obtained with our network on the same validation set. The dashed blue vertical line indicates when the optimal Jaccard index of the classical EM algorithm exceeds the score of our network. The figure on the right involves a logarithmic scale to ease the visualisation of the first part of the graph. In further experiments, we launched 6 training runs of our neural network and we evaluated them on DAVIS2016; we obtained an average score of 68.7 and a variance of 0.41, showing the high stability of our method.

2.4.4 Further analysis and comparison with the classical EM

In this subsection, a quantitative comparison, in term of speed and accuracy, between classical EM and our OFS network is given along with an experimental study on the

dependency of the classical EM to the initialization of the motion model parameters.

First, we elaborate on the iterative parametric motion segmentation based on the classical EM algorithm. More precisely, we study the influence of the initialisation of θ, parameters of the motion models, on the classical EM algorithm performance. We perform this experiment still using quadratic motion models. In doing so, we also highlight the impact of the neural network alone on the performance of our method.

Even if EM algorithm is guaranteed to converge to a local optimum [89], the quality of the optimum reached is highly dependent on the initialisation of θ. We take as conditional likelihood $p(f_i, z_i^k|\theta_k)$ the Gaussian distribution, which leads to the squared L_2 once applying the log function. We evaluate the classical EM on the DAVIS2016 benchmark. Let us recall that the performance criterion is the Jaccard index computed on the primary moving object, not the value of the likelihood being maximised.

To emphasize the dependence of the classical EM on the initialization of θ, we study the evolution of the Jaccard index over the validation set with respect to the number of available random initializations. For each point of the curve, we try available initialisations, and select for each frame the one that maximizes the likelihood. We repeat a second time this experiment, but now using the Jaccard index as selection criterion. Of course, this optimal Jaccard index remains speculative in practice, since we need the ground truth to find the best Jaccard score for each sample. Let us stress that the computational cost involved is enormous in both cases.

We summarize the results obtained on the DAVIS2016 validation set in Fig.2.4. We observe that the optimal Jaccard index for the classical EM algorithm increases with the number of initialisations until reaching a plateau where testing additional initialisation does not improve the results. To ensure this behaviour, we extended this experiment up to a huge number of 3000 initialisations. The optimal (speculative) Jaccard index for the classical EM goes up to 72.3. The Jaccard index for the classical EM when selecting the initialisation with the maximum likelihood, the only means available in practice without ground-truth, tops out at 63.7 . This demonstrates the impact of the initialisation step on the performance of the classical EM algorithm. As mentioned above, the optimal Jaccard index is only speculative. Thus, we consider this score of 72.3 as a gold standard.

Importantly, Fig.2.4 shows that the performance of our network is relatively close to the gold standard given by the speculative optimal Jaccard index of the classical EM algorithm. Besides, the latter needs a number of 41 initialisations to exceed the score of our network. This is already a large number, since, even with our efficient GPU

	Classical EM		Our network
Number of inferences	1	500	1
Average Jaccard (DAVIS2016)	59.0 ± 0.35	63.5	69.3
Inference time / frame (in ms)	9.6	4800	8
Number of parameters	W × H + 12 × K		497 millions + 66 × K
Nb. param. (for K=2,W=128,H=224)	28696		497 millions

Table 2.4 – Quantitative comparaison between classical EM and our trained network on several aspects. Inference time is computed on Tesla-V100 in all cases. Accuracy (average Jaccard) is evaluated on DAVIS2016 validation set. The size of the input flow is noted (W,H) and the number of layers K. Chapitre3/figures for the classical EM are given for two configurations: 1) just applying it once for a random initialisation, but for the accuracy score and a fair comparison, we prefer to give the mean and standard deviation of the average Jaccard index estimated over 3000 initialisations; 2) selecting the best initialisation on every frame among 500 initialisations.

implementation, it takes 25 minutes to run the 41 initialisations per sample over the DAVIS2016 validation set on a Tesla P100.

In Fig.2.4, we also observe that our network outperforms by a large margin the classical EM algorithm when the initializations are selected according to the highest likelihood, even for many tested initialisations, the only means available in practice without ground-truth. In addition, the network does not need motion models, and of course, does not involve any initialisation on motion parameters θ, when segmenting the optical flow. Moreover, we can easily adopt robust loss functions for the network training instead of the squared L_2 loss.

In Table 2.4, we report a quantitative comparison based on several aspects between our method and the classical EM. Accuracy and inference time were computed on the DAVIS2016 validation dataset. As we mentioned above, results using the classical EM strongly depend on the initialisation of the motion parameters. To assess the accuracy of the classical EM, we report first the average score obtained using randomly chosen motion parameters, and secondly, the score obtained by selecting the best initialization from a set of 500 random initialisations. We used 500 here since this is the number of initialisations where the Jaccard score starts to plateau at its optimal value, as shown in Fig.2.4. By design, our method involves a segmentation network with a very large number of parameters to train and store. However, as demonstrated in Table 2.4, our method exhibits strong advantages compared to the classical EM regarding inference time and outperforms it by a large margin in term of accuracy.

Figure 2.5 – Results obtained with our method for four masks ($K = 4$) regarding multiple motion segmentation. First row: one image of the video. Second row: input optical flow displayed with the HSV color code that is depicted in Fig.2.1. Third row: motion segmentation maps with four masks, one colour per mask (the four masks may be not present if not necessary). We adopt the same color code for all segmentation maps (dark blue: mask 1, light blue: mask 2, green: mask 3, yellow: mask 4). Examples are drawn from DAVIS2016, FBMS59 and SegTrackV2 datasets. Videos in lexicographic order: *mallard-fly, hockey, libby, swing, hummingbird, cars5, cars4, people2.*

2.4.5 Multiple motion segmentation

	Davis2016 ($\mathcal{J}$)	SegTrack V2 ($\mathcal{J}$)	FBMS59 ($\mathcal{J}$)	MoCA ($\mathcal{J}$)
$K = 3$	75.1	55.0	62.1	64.4
$K = 4$	76.0	59.6	64.7	66.8
$K = 5$	77.2	59.7	65.1	66.6
$K = 6$	78.3	62.0	66.0	68.0
$K = 7$	78.3	62.8	66.9	67.5

Table 2.5 – Results on DAVIS2016 validation dataset, SegTrackV2, FBMS59 validation dataset and MoCA obtained with our method for several numbers of masks. Quantitative evaluation is performed here using ground truth for selecting the right masks as described in the main text.

Our method can handle by design the segmentation of multiple motions. Indeed, it has been defined from the start with K masks. In Section 2.4.2, we took into account only two masks for the evaluation on VOS datasets, because the challenge and the ground-truth have been defined in this way. In this subsection, we report additional experiments with several masks ($K \in \{3, 4, 5, 6, 7\}$) in Table 2.5. Visual results are also displayed in Fig.4.7 for $K = 4$. They were obtained on videos from DAVIS2016, FBMS59 and SegTrack-V2.

We observe that our method can deal with multiple motions in the video and correctly segment them. This figure includes several examples of articulated motion (e.g., mallard-

fly, hockey), and independently moving objects (e.g., cars5, cars4, people2). The figure also contains examples corresponding to “virtual" failure cases encountered in the two-mask VOS challenge reported in Section 2.4.2. We mean segments that really correspond to moving objects but are not included in the ground truth that is focused on the primary moving object. These results demonstrate that, when involving more than two masks, we can correctly deal with interfering motions such as motion parallax (e.g., libby, swing) or additional moving objects (e.g., cars5). In addition, our multimask parametric motion segmentation could deal with objects comprising several significant planar surfaces of distinct orientation, each of them corresponding then to different optical flow segments.

We evaluated our multi-mask segmentation on the VOS datasets presented in Section 4.4. As we generate now several masks, segment selection is an issue for computing the $\mathcal{J}$ score that is based on two masks (foreground moving object versus background). Just to be able to compute this score and to figure out the performance we could reach (at least an upper bound), we declare as foreground all layers that overlap for their most part the ground-truth foreground mask. Scores are collected in Table 2.5. As expected, performance increases with the number of masks up to a certain point. We observe that we get very high scores on the four benchmarks, and, for $K > 3$, we outperform all the other comparable methods by a relatively large margin (see Table 3.2). We are even close to the performance of supervised methods.

Of course, this would not be a valid procedure to extract the primary moving object in practice, since we need the ground-truth masks. Here, it is just used to consider a quantitative evaluation on the benchmarks. It gives an insight into the quality of our multiple motion segmentation, since, so doing, we are not modifying our segmentation masks, but only selecting them. It also highlights the potential of our multiple motion segmentation method, if paired with an effective mask selection procedure dedicated to the considered downstream task.

Based on visual assessment of our results, we also observed that our method implicitly ensures temporal consistency to some extent. Indeed, motion segments are consistently segmented over time. Moreover, they generally correspond to the same mask number within the same video. This seems to be particularly true for the background when looking at the segmentation results. We illustrate this behaviour in Fig.2.6. This is an appealing property, since the output of the network could be easily exploiting for tracking or for higher-level dynamic scene understanding.

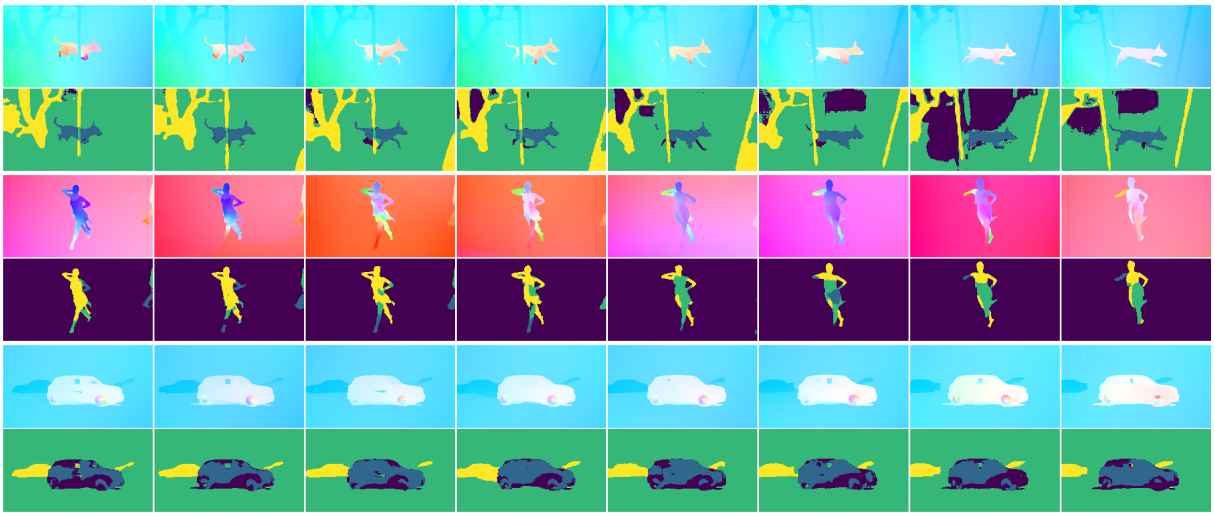


Figure 2.6 – Illustration of the implicit temporal consistency ensured (to some extent) by our method with four masks. For each group, first row: input optical flow fields displayed with the HSV color code, second row: OFS maps. The color code is the same for all segmentation maps (dark blue: mask 1, light blue: mask 2, green: mask 3, yellow: mask 4). Examples are drawn from DAVIS2016 (from top to bottom): *libby, dance-twirl*, and *car-roundabout* videos.

2.5 Discussion

Our work takes its origin in the classical methods of image motion segmentation. The main idea is to group elements of similar nature under a common set of parameters. The novelty of CNN approaches is to train a model that extracts groups without relying on an iterative process at inference. In addition, they engage by design an implicit consistency making the network robust to corrupted observations.

As discussed in [47], region (or layer)-based motion segmentation approaches face the important challenge of specifying a proper model (or regularization) representing the flow within each segment. A too weak regularization may make the flow reconstruction trivial (i.e., the reconstruction is just a copy of the input flow). Conversely, a too strongly regularized model may make the flow reconstruction coarse, since it is likely to be not expressive enough.

Hereafter, we compare our design choices with the ones of the two competing unsupervised optical-flow segmentation methods: CIS [47] and MoSeg [46]. They rely on a different view of the problem. CIS circumvents the regularization problem by inpainting the flow of the foreground region from the flow of the background one and vice versa. In

this setting, any model, even unconstrained, could not correctly inpaint the flow in the event of a perfect segmentation, thus alleviating the problem of regularization. MoSeg addresses the regularization problem by employing a representational bottleneck linked to the architectural structure of the network. Indeed, they force the flow to be reconstructed only from two latent representations of reduced size, one for the foreground and the other one for the background.

On our side, we address the regularization problem by using polynomial motion models. Of course, parametric motion models may lack expressiveness. However, we claim that it is a well-funded choice for the task of motion segmentation. Indeed, there are clear relationships between 2D parametric motion models in the image and 3D motion in the viewed scene [30]. For instance, the 8-parameter quadratic motion model corresponds to the projection into the image of the rigid motion of a planar surface. Relying on those simple parametric motion models, our method achieves excellent performance as demonstrated in Table 3.2.

Compared to CIS, our method bypasses the difficulty to train a complicated inpainting generative model. Compared to MoSeg, it is not attached to a dedicated architectural structure and enables light and efficient architectures for motion segmentation. In addition, the use of parametric motion models for flow representation and of a simple network structure makes the training easier. We advocate that this choice allows for a better generalisation. We consider this point as a critical feature in real-world applications and highlighted it in our evaluation protocol (Section 2.4.2).

Additionally, CIS and MoSeg methods are devoted to a two-mask configuration (moving object *vs.* background), whereas our method can deal with multiple motions by design. Indeed, the use of parametric motion models allows us to perform multilayer motion segmentation (Fig.4.7, Table 2.5), with for example the segmentation of the background moving objects and of articulated motions.

2.6 Partial Conclusion

We have defined an original, real-time, unsupervised method for motion segmentation taking optical flow as input. We leveraged the EM paradigm to define a mathematically well-founded loss function and the training stage of our neural network. No manual annotation is required. We have also designed an effective and simple data augmentation scheme adapted to optical flow fields. In contrast to the classical EM algorithm, our

method is not iterative at the inference stage and does not need to estimate parametric motion models at test time. In addition, our OFS method can handle by design the segmentation of multiple motions.

Our method outperforms state-of-the-art comparable unsupervised methods on the DAVIS2016 and FBMS59 benchmarks, and is the second best for SegTrackV2 and MoCA, yielding the best overall performance. Additionally, the version with more than two masks opens a promising perspective on this point. Interestingly, our OFS method implicitly provides rather time-consistent segments.

In the two next chapters, we will explicitly and deeply handle the temporal dimension of the motion segmentation problem. For the sake of brevity, we will call from now our EM-driven method the EM method.

Chapter 3

Temporally-consistent Segmentation of Multiple Motions

Temporal consistency is a key feature of motion segmentation, but it is often neglected. In this chapter, we propose an original unsupervised spatio-temporal framework for motion segmentation from optical flow that fully investigates the temporal dimension of the problem. More specifically, we have defined a 3D network for multiple motion segmentation that takes as input a sub-volume of successive optical flows and delivers accordingly a sub-volume of coherent segmentation maps. In this chapter, we start with a short-term temporally-consistent motion segementation, meaning that the input sub-volume remains limited (typically, three optical flow fields). The next chapter, Chapter 4, will deal with long-term temporally-consistent motion segmentation.

Our network is still trained in a fully unsupervised way, and the loss function combines a flow reconstruction term involving spatio-temporal parametric motion models, and a regularization term enforcing temporal consistency on the masks. We have specified an easy temporal linkage of the predicted segments. Besides, we have proposed a flexible and efficient way of coding U-nets.

3.1 Temporal Dimension of the Problem

Clearly, the motion segmentation problem has a strong temporal dimension, as motion is generally consistent throughout the video, at least in part of it within video shots. The use of one optical flow field at each given time instant may be sufficient to get the segmentation at frame t of the video. However, extending the temporal processing window can be beneficial. Introducing temporal consistency in the motion segmentation framework is certainly useful from an algorithmic perspective: it may allow to correct local errors or to predict the segmentation map at the next time instant. Beyond that, temporal consistency is an intrinsic property of motion that is essential to involve in the

formulation of the motion segmentation problem.

In this chapter, we propose an original method for multiple motion segmentation from optical flow, exhibiting temporal consistency, while ensuring accuracy and robustness. To the best of our knowledge, our optical flow segmentation (OFS) method is the first one to involve short- and long-term temporal consistency. We are considering a fully unsupervised method, which overcomes tedious or even unfeasible manual annotation and provides a better generalization power to any type of video sequences.

The main contributions of our work are as follows. We adopt an explicit space-time approach. More specifically, our network takes as input a sub-volume of successive optical flows and delivers accordingly a sub-volume of coherent segmentation maps. Our network is trained in a completely unsupervised manner, without any manual annotation or ground truth data of any kind. The loss function combines a flow reconstruction term involving spatio-temporal parametric motion models defined over the flow sub-volume, and a regularization term enforcing temporal consistency on the masks of the sub-volume. Our method also introduces a latent represention of each segment motion and enables an easy temporal linkage between predictions. In addition, we have designed a flexible and efficient coding of U-nets.

The rest of the chapter is organized as follows. Section 3.2 is devoted to related work specifically concerned with the temporal dimension. In Section 4.2, we describe our unsupervised 3D network for multiple motion segmentation embedding temporal consistency. Section 4.3 collects details on our implementation. In Section 4.4, we report results on several VOS benchmarks with a comparison to several existing methods.

3.2 Additional Related Work

Related work on motion segmentation was globally addressed in Chapter 1. Here, we add comments on references specifically concerned with the temporal dimension.

The temporal dimension of the motion segmentation problem has been somewhat considered in various ways. First, regarding classical approaches, in [31] the motion partition at time t was predicted from the one obtained at time $t-1$ using the affine motion models estimated between images $t-1$ and t for each segment, within a robust MRF-based method. The authors in [105] showed that it was beneficial to introduce temporal layer constancy over several frames to perform motion segmentation in a MRF-based and graph-cut optimization framework. In [51], large time windows were taken into

account, allowing the use of point trajectories within a spectral clustering method but resulting in sparse displacement fields.

More recently, regarding deep learning approaches, segmentation at a given time instant t is enforced during the training phase of the network designed in [46], by considering several time pairs involving instants before and after t and their corresponding optical flow fields. In [25], the scope is a bit different, since the authors deal with amodal segmentation, i.e., the recovery of the whole object even in case of occlusion or temporary static state. To this end, they introduce a multi-frame analysis comprising a transformer encoder, while using synthetic ground truth for training involving human annotation. In the same vein, multiple object segmentation is addressed in [26] with the addition of depth-ordered layer representation. A self-supervised model for VOS has been very recently proposed in [106], taking several consecutive RGB frames as input. Optical flow is computed at training time. Furthermore, a temporal consistency term is added in the loss function. However, the temporal consistency is not applied to two consecutive segmentation maps, but for different pairings between frame t and another (more or less distant) frame. In [107], temporal feature propagation is an important component of the framework of spatio-temporal transformers designed for video object segmentation.

Another way to integrate the evolution of the video is to involve memory modules, as in [16] where a two-stream neural network, encompassing spatial and temporal features, is equipped with an explicit memory designed with convolutional gated recurrent units. Memory networks that can be trained end-to-end are leveraged in [108] for semi-supervised VOS. The memory is fed by frames with object masks and can be dynamically updated. Memory is introduced through a recurrent network for zero-shot VOS in [109]. It is fully end-to-end trainable and involves both the spatial and temporal domains.

Video prediction is a topic of growing importance surveyed in [110], which we can also mention in this discussion. The objective is to forecast future frames of a video sequence from several past frames. The authors of [111] concentrate on the prediction of the transformations between successive images, represented by affine models, to generate the next frame of the sequence. In [112], the proposed framework handles in different ways predictable moving regions and disoccluded regions, from a confidence factor evaluated after warping. The latter regions are predicted by a dedicated inpainting network. Motion related to actions is predicted from previous frames in [113] in the context of robot interactions, which enables to be partially invariant to object appearance. In [114], convolutional features of the Mask R-CNN instance segmentation model are predicted to

produce the segmentation of future frames. Other works proposed LSTM networks for semantic segmentation [115], and local frequency domain transformer networks [116]. In a different perspective, the prediction of probable motion patterns is used at the training stage in [3], as a cue to learn objectness from videos.

Our fully unsupervised approach differs from all these previous works in several respects. We rely only on optical flow and take a sub-volume of OF fields as input, providing a sub-volume of consistent segmentation maps. Moreover, we introduce space-time parametric motion models and temporal consistency between consecutive masks.

3.3 Temporally-consistent Motion Segmentation Method

We have designed a 3D convolutional network for multiple motion segmentation from optical flow. The network takes a sub-volume of several successive optical flow fields as input. Temporal consistency is expressed in two main ways at the training stage. Firstly, we introduce space-time parametric motion models to represent the flow in each segment over the space-time sub-volume. Secondly, we define a regularization term in the loss function enforcing stable labeling of the motion segments over the sub-volume. We call ST-MS this new motion segmentation method.

3.3.1 Spatio-temporal parametric motion model

For each motion segment k from a set of K segments, we define a spatio-temporal parametric motion model $\tilde{f}_{\theta_k,\alpha_k}$ in the (x, y, t) volume of the sequence, introducing a temporal variation of each spatial parameter of the model. For instance, if we consider an affine spatio-temporal motion model, it is given by:

$$\begin{aligned}\tilde{f}_{\theta_k,\alpha_k}(x,y,t) = (&\theta_{k_1} + \alpha_{k_1}t + (\theta_{k_2} + \alpha_{k_2}t)x + (\theta_{k_3} + \alpha_{k_3}t)y,\\ &\theta_{k_4} + \alpha_{k_4}t + (\theta_{k_5} + \alpha_{k_5}t)x + (\theta_{k_6} + \alpha_{k_6}t)y)^T,\end{aligned} \tag{3.1}$$

where $\theta_k = (\theta_{k_1}, .., \theta_{k_6})$ corresponds to the spatial part of the motion model, and $\alpha_k = (\alpha_{k_1}, .., \alpha_{k_6})$ its temporal extension to account for possible variations of the flow over time.

By developing eq.(3.1), an equivalent formulation is:

$$\tilde{f}_{\theta_k,\alpha_k}(x,y,t) = (\theta_{k_1} + \theta_{k_2}x + \theta_{k_3}y + \alpha_{k_1}t + \alpha_{k_2}xt + \alpha_{k_3}yt, \\ \theta_{k_4} + \theta_{k_5}x + \theta_{k_6}y + \alpha_{k_4}t + \alpha_{k_5}xt + \alpha_{k_6}yt)^T. \quad (3.2)$$

Similar expressions can be straightforwardly defined for any spatio-temporal quadratic motion model. For the full quadratic motion model with 12 parameters, we get:

$$\begin{aligned}\tilde{f}_{\theta_k,\alpha_k}(x,y,t) = \\ &(\theta_{k_1} + \alpha_{k_1}t + (\theta_{k_2} + \alpha_{k_2}t)x + (\theta_{k_3} + \alpha_{k_3}t)y + \\ &(\theta_{k_4} + \alpha_{k_4}t)x^2 + (\theta_{k_5} + \alpha_{k_5}t)xy + (\theta_{k_6} + \alpha_{k_6}t)y^2, \\ &\theta_{k_7} + \alpha_{k_7}t + (\theta_{k_8} + \alpha_{k_8}t)x + (\theta_{k_9} + \alpha_{k_9}t)y + \\ &(\theta_{k_{10}} + \alpha_{k_{10}}t)x^2 + (\theta_{k_{11}} + \alpha_{k_{11}}t)xy + (\theta_{k_{12}} + \alpha_{k_{12}}t)y^2)^T. \end{aligned} \quad (3.3)$$

The spatio-temporal motion model encompasses the successive locations of segment k in the sub-volume. In practice, we take a sub-volume of three flow fields sampled every τ time instants, i.e., at $t-\tau, t$, and $t+\tau$. A typical value for τ is 1, that is, a triplet of three consecutive flows, but, other values can be chosen. If τ is positive, respectively negative, it means that we proceed forward, respectively backward, in time. We can cope with a single τ value, or with several ones jointly, all the triplets being centered on t. Other types of sub-volumes could be handled as well.

3.3.2 Network architecture

The overall principle of our unsupervised 3D multiple motion segmentation network is illustrated in Fig.3.1. It is based on the U-net architecture [86]. The network takes a sub-volume of flow fields as input around time t and jointly predicts the sub-volume of segmentation maps, while enforcing temporal consistency on them. One possibility is to keep only the segmentation map m_t, and the process is performed again at the next time instant $t+1$. However, alternatives can be considered, as keeping the three maps.

3.3.3 Loss function

The loss function of our 3D motion segmentation network is composed of two terms: a segment-wise flow reconstruction term and a temporal consistency one on the predictions.

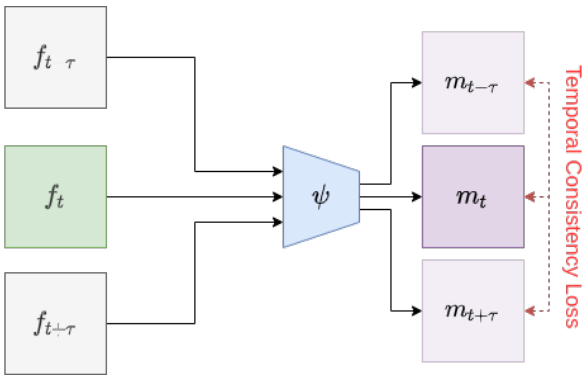


Figure 3.1 – Principle diagram of our space-time multiple motion segmentation network ψ, taking as input a space-time sub-volume composed of three flow fields sampled every τ time instants and delivering a sub-volume of three coherent segmentation maps.

The first term, denoted $\mathcal{L}_r$, expresses how the estimated parametric motion models fit the input optical flow within each segment. It writes:

$$\mathcal{L}_r = \frac{1}{3} \sum_{s \in \{-\tau, 0, \tau\}} \frac{\sum_{i \in \mathcal{I}} \sum_{k=1}^{K} m_k(i, t+s) \| f(i, t+s) - \tilde{f}_{\theta_k, \alpha_k}(i, t+s) \|_1}{\sum_{i \in \mathcal{I}} \| f(i, t+s) \|_1}, \tag{3.4}$$

where $i = (x, y)$ is a site of the image grid $\mathcal{I}$, K is the number of motion layers or segments, $f(i, t)$ is the flow vector at site i and time instant t, and $m_k(i, t)$ denotes the probability of site i to belong to motion segment k at time t, that is, the prediction (or output) of the motion segmentation network. f_t will designate the optical flow field at time t, $f_t = \{f(i, t), i \in \mathcal{I}\}$. We use the robust norm L_1 to overcome the presence of outliers in the motion segment, especially at the beginning of the training when segments are not yet well extracted, and to mitigate possibly wrong flow vectors, around motion discontinuities in particular.

The second term, denoted $\mathcal{L}_c$, enforces temporal consistency of the motion segments. To do this, the probability of site i to belong to segment k is assumed to be stable over time within the considered triplet. We have:

$$\mathcal{L}_c = \frac{1}{2K|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{k=1}^{K} \big(|m_k(i, t-\tau) - m_k(i, t)| + |m_k(i, t) - m_k(i, t+\tau)| \big), \tag{3.5}$$

where $|\mathcal{I}|$ designates the number of sites over the image grid $\mathcal{I}$. For the sake of simplicity,

we have adopted an Eulerian standpoint [117], that is, we compare segment labels over time at any given site i of the image grid. In fact, every point is likely to move and a Lagrangian standpoint [117] would be more appropriate. It would require the use of the optical flow vectors to track every point over time. However, the computed flow field may be imprecise or even erroneous at some points, and besides, interpolation operations would be necessary since the flow components take on real values. The Eulerian temporal constraint does not mean that the site should be static within the triplets, but it works as long as the site lies on the overlap of the successive positions of the moving parts. However, it does not make sense on occluded or disoccluded parts. This justifies the use of the L_1 norm to deal with the latter configuration.

We further prevent from enforcing the temporal consistency over occlusion areas by ignoring sites i in the summation over $\mathcal{I}$ in eq.(4.6) that exhibit a large temporal flow difference. More precisely, we set a threshold λ so that a quantile η of sites i is discarded as follows:

$$p(\|f(i,t+\tau) - f(i,t)\|_1 \geq \lambda) \leq \eta. \tag{3.6}$$

In practice, we take $\eta = 1\%$. In doing so, we make an implicit assumption on the overall surface of the occlusion areas, but it seems reasonable for the datasets we deal with. The loss function is the sum of the two terms:

$$\mathcal{L} = \mathcal{L}_r + \beta \mathcal{L}_c. \tag{3.7}$$

We simply set $\beta = 1$, since the two terms of the loss function are properly normalized.

With our approach, we can easily infer a temporal linkage between successive predictions, as explained below.

3.3.4 Principle of segment selection for evaluation

To evaluate our method and compare it with similar unsupervised methods, we use VOS benchmarks as a substitute for optical flow segmentation (OFS) benchmarks, as no such benchmarks are available. The two tasks are close. However, the VOS one is attached to the notion of a primary object of interest moving in the foreground (sometimes, a couple of objects). As a consequence, we have to select the right segments to cope with the binary ground truth of the VOS benchmarks, as described below.

First, we link throughout the video the K segments obtained at each instant t. The

fact that we proceed with sub-volumes imposing common segment labels within the sub-volume, and that consecutive sub-volumes share two masks, helps us establish the temporal linkage. We link segments from one time instant to the next one, using the IoU measure (intersection over union) as linkage criterion. This is achieved throughout the video by sub-sequences, and then, the comparison of the K segments with the ground truth is done at this sub-sequence level, enforcing the temporal dimension of our approach. A detailed description on how to link predictions and select the segments for evaluation is provided in the next subsection.

This procedure is applied on the three datasets DAVIS2016, SegTrackV2 and FBMS59. In practice, we take sub-sequences of 10 frames. The segment association, required to compute the Jaccard score, leverages the ground truth, but with the notable fact that it is done only once at the sub-sequence level, which shows the ability of our method to supply long-term stability.

Let us mention that we are able to infer another information related to motion. We can generate a latent representation of the segment motion. More specifically, this latent representation $\chi(S_i)$ of segment S_i is defined as the average value of the latent vectors, normalized in mean and variance, of the segment sites. In future work, we could establish a motion similarity measure between two segments S_i and S_j, given by the dot product of their respective latent representations χ_j and χ_j. This motion similarity could be used in the temporal linkage of the segments in addition to IoU, or to merge segments, especially if we take a large value for the mask number K. More details on the different items presented in this subsection are provided in Appendix 7.2.

3.3.5 Linking predictions and selecting segments

We detail the procedure for linking predictions in a subsequence and for selecting optimal segments for evaluation. Subsequences are composed of $T+1$ consecutive flow fields. Let us define:

$$\breve{m}_t(i) = \arg \max_{k=1,..,K} m_k(i,t), \tag{3.8}$$

where $m_k(i,t)$ is the probability of site i to belong to segment k at time t. Let $\breve{m}_t$ be the segmentation map at time t encompassing the (up to) K segments predicted by the network, $\breve{m}_t = \{\breve{m}_t(i), i \in \mathcal{I}\}$. In other words, $\breve{m}_t$ is the label array representing the set of segments extracted at time t, segments being (non necessary connected) layers. We will also use the term mask to designate $\breve{m}_t$, when no confusion can occur. The prediction of

the network is given for a triplet of input flow fields $(f_{t-1}; f_t; f_{t+1})$ as a triplet of masks $(\breve{m}_{t-1}; \breve{m}_t; \breve{m}_{t+1})$ for $\tau = 1$. All those triplet predictions are produced in parallel and the triplet output are independent as illustrated in Fig.3.2.

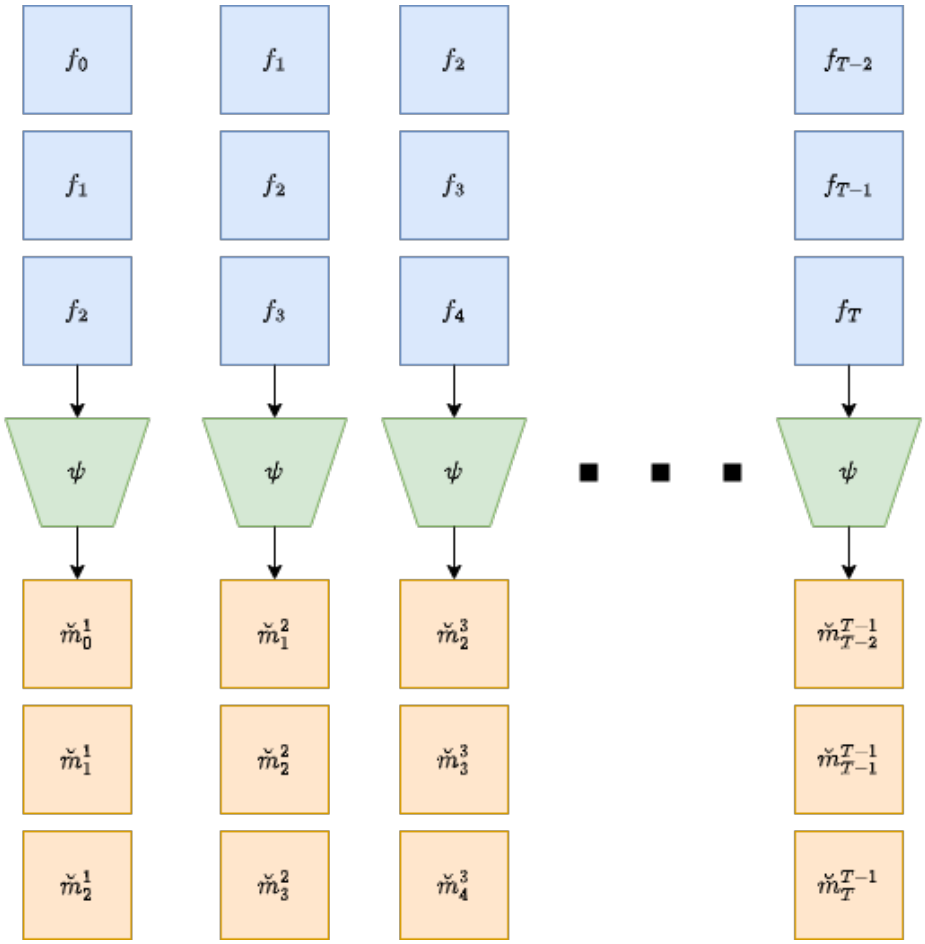


Figure 3.2 – Output of the network by triplet (for $\tau = 1$) within a subsequence covering the time interval $[0, T]$. The lower index t of each mask $\breve{m}_t^{t'}$ represents the time instant of the corresponding flow field f_t, and the upper one t' corresponds to the time instant when it is produced, that is, the one of the reference (central) flow field of the triplet. Segmentation masks with the same lower index correspond to different segmentation instances of the same flow field.

Linking predictions

The masks in the same triplet are sharing common labels but not necessary across triplets. By label, we mean mask number. We have to link the labels by finding correspondences between triplets. Since we have three versions of the same mask $\breve{m}_t$, it is

straightforward to achieve it.

First, we need to introduce an additional notation $\breve{m}_t^{t'}$ as defined below. The segmentation mask $\breve{m}_t^{t'}$ ($t' \in \{t-1, t, t+1\}$) of width W and height H, consisting of K non-overlapping classes ($\breve{m}_t^{t'} \in \{1, .., K\}^{W \times H}$) and corresponding to flow f_t, is the one predicted by the network when it takes a triplet centered around t' as input. We have to find the best label association between instances $\breve{m}_t^{t'}$ of the same mask. To do that, we compute a label reassignment table that will be applied to the two masks $\breve{m}_t^t$ and $\breve{m}_{t-1}^t$. The reassigned label l^* for each label $l \in \{1, .., K\}$ is given by:

$$l^* = \arg \max_{k \in \{1,..,K\}} J(k_t^{t-1}, l_t^t) + J(k_{t-1}^{t-1}, l_{t-1}^t), \tag{3.9}$$

where J is the IoU score between two segments. The binary array $k_t^{t'}$ is defined as follows:

$$k_t^{t'}(i) = \begin{cases} 1 & \text{if } \breve{m}_t^{t'}(i) = k \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

$l_t^{t'}$ is defined in a similar way. We proceed by pairs, since two consecutive triplets share two masks as illustrated in Fig.3.3 (the pairs of arrows). Starting from $t = 0$, we propagate the labels to the whole subsequence using the criterion of eq.(3.9). After the label reassignment propagation, we rename each $\breve{m}_t^{t'}$ as $\bar{m}_t^{t'}$.

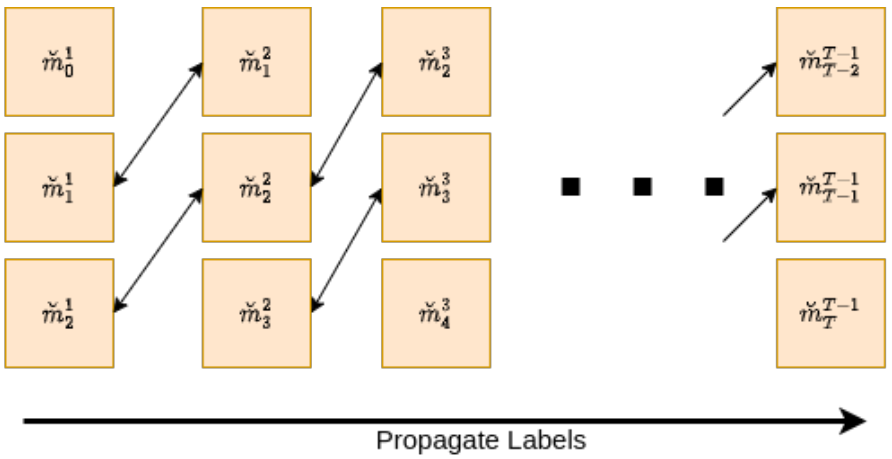


Figure 3.3 – Propagation of the labels (i.e., masks numbers) over the subsequence.

Select optimal labels

We select an optimal set of segments for evaluation, based on the ground truth, for the entire subsequence. The goal is to show that, since we have coherent labels within the subsequence, we can select the optimal segments at the subsequence level. First, we unroll our sequence by only keeping the central prediction for each step from $t = 1$ to $t = T - 1$, and just retrieving the single instance produced for the first ($t = 0$) and last ($t = T$) time steps as depicted in Fig.3.4.

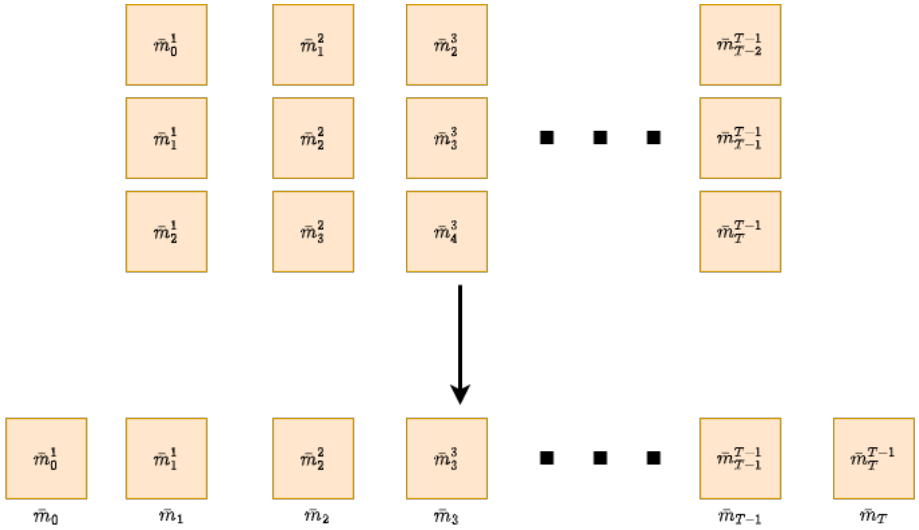


Figure 3.4 – Formation of the mask series $\{\bar{m}_t, t = 0, .., T\}$ over the subsequence for evaluation , once the label propogation step is achieved.

Then, for the entire subsequence, we select the subset $\mathcal{S}^*$ of labels (that is, of segments) to constitute the predicted foreground, using the binary ground-truth g_t. The selected label subset $\mathcal{S}^*$ is given by:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subset \mathcal{P}(\{1,..,K\})} \sum_{t=0}^{T} J(\bigcup_{l \in \mathcal{S}} \bar{l}_t, g_t), \tag{3.11}$$

where $\mathcal{P}(\{1, .., K\})$ is the partition of the labels in the subsequence, and the binary masks $\bar{l}_t$ correspond to the label mask $\bar{m}_t$. Once we have selected the subset $\mathcal{S}^*$ of labels, we can use it to build our binary segmentation $\{s_t, t = 0, .., T\}$ on the whole subsequence for evaluation, as illustrated in Fig.3.5.

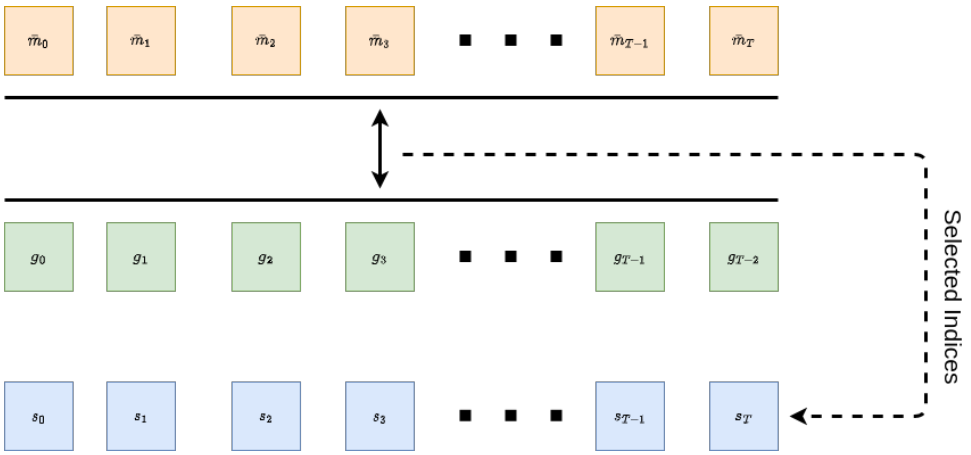


Figure 3.5 – Comparison of the selected segments with the ground truth for evaluation.

3.3.6 Impact of subsequence length

In the temporal segment linkage and the segment selection process described above, we take into account a subsequence of length $T + 1$. For the results reported below, we use a subsequence length of 10 frames. However, we can vary the subsequence length to evaluate the robustness of our method to this parameter. All the evaluations regarding the impact of the subsequence length, are produced from the same trained network and initial segmentation. They are plotted in Fig.3.6. Longer subsequences are more challenging since they require a stronger temporal consistency. Also, they can be more impacted by occlusions or flow estimation errors, which can break label propagation.

We can see that our method is robust to the choice of the subsequence length, and that it is performing well on long subsequences as well.

3.4 Implementation

3.4.1 Network coding

In this work, we explored different ways of dealing with the input optical flow volume and especially different interactions between time steps. We also wanted to use a multi-resolution structure to segment large inputs while preserving fine-grain details. In order to easily handle these different options, we developed an original and flexible implementation [1] of a U-net based on the decomposition of its structure into five modules

1. https://github.com/Etienne-Meunier-Inria/GeneralUnet

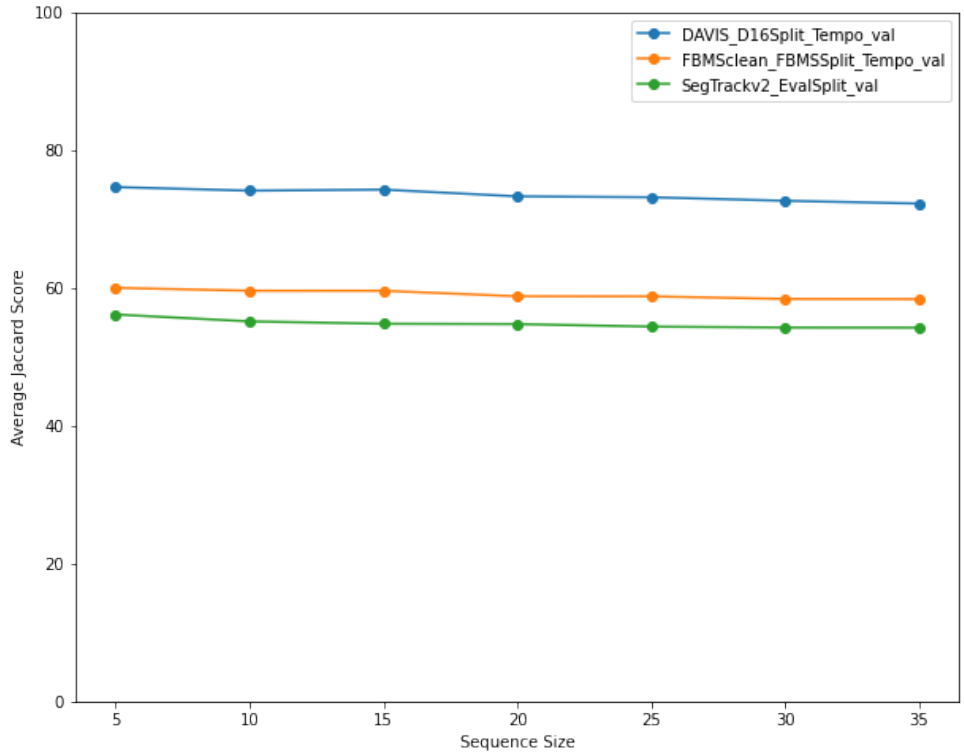


Figure 3.6 – Evaluation of our temporal segment linkage and segment selection process for different subsequence lengths on the three datasets.

as described in Fig.3.7.

The backbone of our network is a version of the classical U-net calling abstract instances of these modules and applying error checks to control their output. Using our implementation, one can easily instantiate a novel architecture by solely implementing desired modules without getting involved in the core steps of the U-net. Since all U-net blocks are composed of the same modules, we can stack them making the code needed to implement a new architecture minimal. The input and bottleneck layers of the U-net are handled seamlessly by using a part of provided modules.

This framework makes it straightforward to implement a multidimensional U-net, to incorporate various transformations in the transit layer (e.g., transformer as in [26] or recurrent CNN as in [115]), or to change the sampling or upsampling steps, keeping the same general skeleton between all these different architectures. For example, in our work, we implemented a version of downsampling and upsampling that is applied only on the spatial dimension, while double convolutions are applied on both the spatial and temporal dimensions. The proposed implementation encompasses many of the solutions described

in Section 3.2, and allows new ideas to be tested quickly and in a standardized way. It is also applicable beyond motion segmentation. We have make our code available in an open source repository [2].

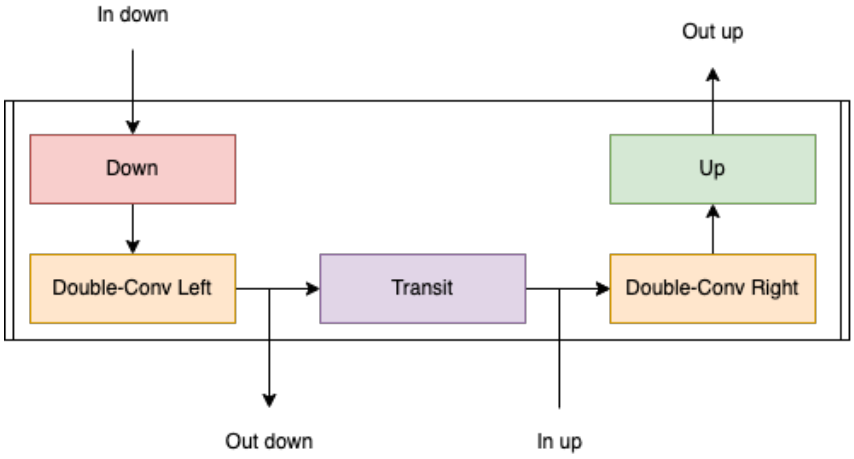


Figure 3.7 – Diagram of the prototype layer in our implementation of the U-net. "Double-Conv Left" and "Double-Conv Right" are applying several transformations of the feature maps after downscaling and before upscaling respectively. In classical U-net architecture, it is a 2D convolution kernel applied on the spatial dimension followed with a batch norm. "Down" is a downscaling layer that reduces the spatial dimension of the feature map. In classical U-net architecture, it is implemented using max pooling 2D. "Up" is a block that increases the dimension of the feature map usually implemented with bilinear interpolation or transposed convolution. "Transit" is the connection between the down path and the up path. In classical U-net architecture, it is a skip connection.

3.4.2 Implementation details

As in [46], [55], we adopt the RAFT method [56] to compute the optical flow fields. More specifically, we use the RAFT implementation [3] with network weights fine-tuned on the MPI Sintel dataset [97]. We downsample them to feed the network with 128×224 vector fields as input. Thus, we achieve much more efficient training and inference stages. The output segmentation maps are then upsampled to the original frame size for evaluation w.r.t. the ground truth.

Regarding the estimation of the spatio-temporal parametric motion model, since the x and y coordinates are normalized within $[0, 1]$, we apply a similar normalization for the t

2. https://github.com/Etienne-Meunier-Inria/GeneraUnet
3. https://github.com/princeton-vl/RAFT

coordinate. For instance, if we set $\tau = 1$, we get as normalized time values: $t-1 = -0.33$, $t = 0$, and $t+1 = 0.33$. We use the full quadratic motion model, with 12 spatial parameters and 12 temporal ones, in all the reported experiments. This type of motion model enables to account for complex depth surfaces and movements.

We use only the prediction $m_k(i,t)$ of the network obtained when considering the flow triplet $(f_{t-\tau}, f_t, f_{t+\tau})$ to decide to which segment k site i belongs to at time instant t. More precisely, we select for each point i the segment $\hat{k}$ with the highest probability. In all the experiments reported in Section 4.4, we simply use a single value for τ, $\tau = 1$. We refer the reader to Appendix 7.2 for possible alternatives. Let us recall that negative values of τ mean that we proceed backward in time. A combinaison of several τ values could also be used .

3.4.3 Data augmentation and network training

We perform two types of data augmentation. The first one consists in adding a global flow to the input flow as done in [55]. The global flow is given by a full spatio-temporal motion model whose parameters are chosen at random. We just make sure that the added global flow is roughly equivalent in magnitude to the initial flow. The same global flow is added to the three flow fields of the input sub-volume. This type of data augmentation allows us to mimic different camera motions, enforcing that the motion segments are independent of it. For the second type of data augmentation, we corrupt one input flow out of the three ones. The idea is to simulate a poorly estimated flow field and to compel the temporal consistency to compensate.

Our motion segmentation method is entirely unsupervised. We do not perform any manual annotation in all the experiments. We train the 3D motion segmentation network on the training set of DAVIS2016, once for all. Moreover, the stopping epoch is selected from the loss function evaluated on the DAVIS2016 validation set. We use Adam optimizer with a learning rate of 10^{-4} to train the 3D network. The estimation of the parameters $\{\theta_k, \alpha_k, k = 1, K\}$ of the motion models is achieved with the Pytorch implementation of L-BGFS [103]. Let us recall that we estimate the parametric motion models only at training time.

Our method is very efficient at test time. For the model (small 3D U-net), the computational time amounts on average to 114 *fps* on a GPU P100. The impact is negligible regarding the number K of masks used since only the final layer is modified, and it is proportional to the frame size $|\mathcal{I}|$.

Network modification	No data augmentation	Spatial quadratic motion model	Loss without $\mathcal{L}_c$	Our full method
DAVIS2016 $\mathcal{J} \uparrow$	70.1	70.5	33.1	73.2
SegTrackV2 $\mathcal{J} \uparrow$	52.3	54.5	24.9	55.0
FBMS59 $\mathcal{J} \uparrow$	50.4	54.5	54.5	59.4

Table 3.1 – Ablation study for three main components of our method evaluated on DAVIS2016 validation set, SegTrackV2 and FBMS59. Only one component is modified at a time.

3.5 Experimental Results

3.5.1 Datasets

We have carried out experiments on three VOS datasets: DAVIS2016 [4] [15], SegTrackV2 [5] [88], and FBMS59 [51].

DAVIS2016 consists of 50 videos (and 3455 frames) that are split in a training set of 30 videos and a validation set of 20 videos. They contain diverse moving objects. Only the primary moving object is annotated in the ground truth. The criteria for evaluation on this dataset are the Jaccard score (denoted $\mathcal{J}$), and the contour accuracy score (denoted $\mathcal{F}$).

SegTrackV2 includes 14 videos (with a total of 1066 annotated frames), and FBMS59 contains 59 videos (720 annotated frames), both involving one moving object but sometimes a couple of moving objects. For FBMS59, we use the 30 sequences of the validation set for evaluation. As done in [46], if there are several moving objects, we group them into a single foreground mask for evaluation.

3.5.2 Ablation study

We have conducted an ablation study to assess three main components of our method, two of them being related to the temporal dimension. We proceeded by modifying only one component at a time. The three network components concerned are: *i)* no use of any data augmentation, *ii)* use of a spatial quadratic motion model per frame instead of the spatio-temporal one, *iii)* specification of the loss function without the consistency term

4. https://davischallenge.org/index.html
5. https://paperswithcode.com/dataset/segtrack-v2-1

Method	Training	Input	DAVIS2016 $\mathcal{J}\uparrow$	DAVIS2016 $\mathcal{F}\uparrow$	SegTrack V2 $\mathcal{J}\uparrow$	FBMS59 $\mathcal{J}\uparrow$
Ours	Unsupervised	Flow	73.2	70.3	55.0	59.4
EM (Chapter 2) [55]			69.3	70.7	55.5	57.8
MoSeg [46]			68.3	66.1	58.6	53.1
FTS [104]			55.8		47.8	47.7
TIS_0 [68]			56.2	45.6	-	-
OCLR* [26] (flow-only)			72.1	-	67.6	65.4
GWM [3]		RGB (Flow in loss)	79.5	-	78.3	77.4
MOD [106]			73.9	-	62.2	61.3
TIS_s [68]		RGB & Flow	62.6	59.6	-	-
CIS - No Post [47]			59.2		45.6	36.8
CIS - With Post [47]			71.5		62.0	63.6
DyStab - Dyn [48]	Supervised Features	Flow	62.4		40.0	49.1
DyStab - Stat&Dyn [48]		RGB & Flow	80.0		73.2	74.2
ARP [27]			76.2	70.6	57.2	59.8
MATNet [19]	Supervised	Flow	82.4	80.7		
COSNet [24]		RGB	80.5	79.5	-	75.6

Table 3.2 – Results obtained with our method ($K = 4$) on DAVIS2016, SegTrackV2, and FBMS59, including comparison with unsupervised and supervised methods (scores from cited articles). The Jaccard index $\mathcal{J}$ expresses the correct overlap (intersection over union) between the extracted segments and the ground truth, while $\mathcal{F}$ focuses on segment boundary accuracy (the higher the better). Performance is assessed by the average score over all samples, for all datasets but DAVIS2016. For the latter, the overall score is given by the average of sequence scores. *OCLR is not a truly unsupervised method since it relies on human-annotated sprites to get realistic shapes in the synthetic data used in the training.

$\mathcal{L}_c$. All the ablation experiments were run on the DAVIS2016 validation set, SegTrackV2 and FBMS59 datasets. Results are collected in Table 7.12. We can observe that the spatio-temporal motion model improves the performance of the method, by taking into account the possible motion evolution within the sub-volume. Above all, the introduction of the temporal consistency term $\mathcal{L}_c$ in the loss function is drastically beneficial. Its impact appears far weaker for FBMS59, but this is due to the evaluation procedure, since the evaluation is performed only every ten frames. The ablation study demonstrates the pivotal role of the two components acting at two levels of temporal consistency in the flow segmentation

Figure 3.8 – Results obtained with our method using four masks ($K = 4$), but the network may not necessarily use all the four masks. Two groups of results. For each group, the first row depicts one image of the video, the second row contains the optical flow input represented with the usual HSV color code, the third row displays motion segments (given by layers that are not necessarily connected) with one colour per segment. Samples are drawn from the different datasets.

3.5.3 Quantitative and comparative evaluation

We report in Table 3.2 the results obtained by our method on the three datasets DAVIS2016, SegTrackV2 and FBMS59, along with those obtained by other existing methods. Since our method is fully unsupervised and only uses optical flow as input, we focus on similar methods for a fair comparison. We consider the method categories that we proposed in [55] regarding input and training, by the way very close to other propositions. Our previous motion segmentation method, presented in Chapter 2, will be called EM method from now. We have added a category w.r.t. the network input for two very recent methods, [3], [106], that only use RGB images as imput at test time, the optical flow being only involved in the loss function. Additionally, the OCLR method [26] exploits human-annotated sprites to generate realistic shapes in the synthetic data used in the training. We consider the OCLR version taking only optical flow as input. The post-processing added to the CIS method [47], based on Conditional Random Fields (CRF), is an heavy one, which leads most authors to retain only the version without post-processing for a fair comparison.

Overall, our method shows convincing performance w.r.t. comparable methods, namely,

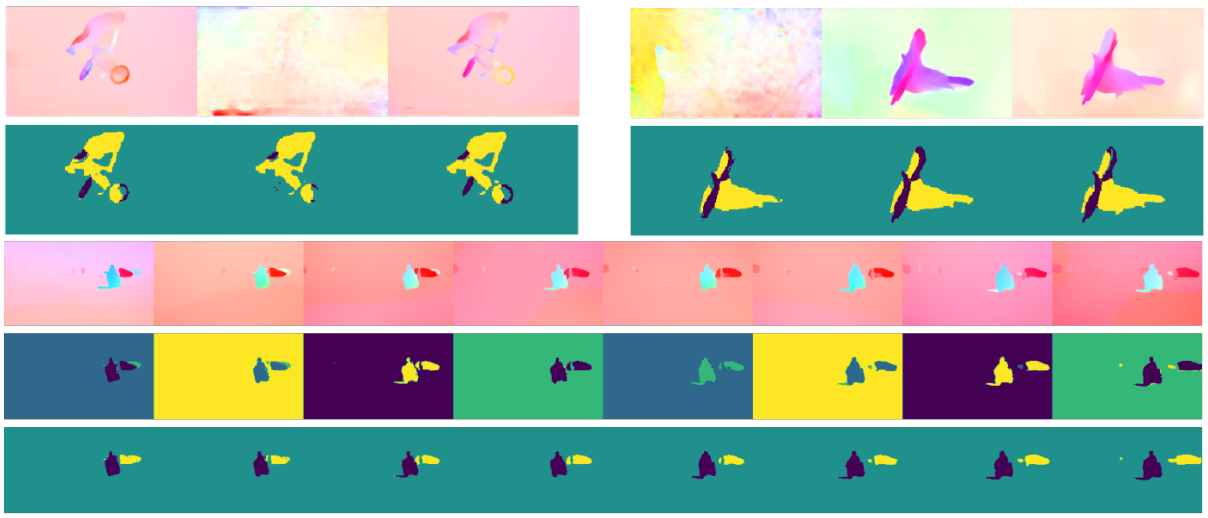


Figure 3.9 – Impact of temporal consistency on several situations with our method ($K = 4$). Two groups of results from top to bottom. First group: two cases of amodal segmentation, in fact corresponding to a repeated image in the video file mimicking a stop (first row contains optical flow input, second row displays motion segments). Second group: one case exemplifying the action of the temporal consistency term $\mathcal{L}_c$ of the loss function (short term) and of the temporal linkage (long term) to maintain the same mask labels over time (first row contains optical flow input, second row displays motion segments obtained without $\mathcal{L}_c$, third row includes motion segments obtained with $\mathcal{L}_c$ and temporal linkage).

unsupervised methods taking optical flow as input. Temporal consistency was properly handled by our method and gave quite satisfying results. More specifically, our method shows an excellent performance on DAVIS2016 and a very good performance on FBMS59. Regarding SegTrackV2, this dataset includes sequences filmed with a poorly controlled handheld camera, which leads to unstable sequences where the contribution of our method cannot be as significant. In addition, information is provided in Appendix 7.3 on the repeatability issue.

3.5.4 Qualitative visual evaluation

Fig.4.7 contains several visual results to demonstrate how our method behaves on different situations. We display result samples obtained on different videos of the benchmarks. We can observe that the segmentation are globally accurate. Since our method can involve K masks, we can properly handle articulated motions, or the presence of several moving objects in the scene, as illustrated in Fig.4.7. We must keep in mind that our

actual target is the OFS task, even if we evaluate our method on VOS benchmarks. Since the VOS benchmarks mainly deal with the segmentation of one primary object moving in the foreground, it may occur some discrepancies with OFS. For instance, the segmentation of additional parts w.r.t. VOS ground truth makes nonetheless sense from the OFS standpoint. Let us mention the cases of a moving car in the background, two animals running, ripples on the water, motion parallax due to static objects in the foreground, as illustrated in several examples of Fig.4.7. It can affect the overall scores reported in Table 3.2.

We gather in Fig.3.9 several result samples that demonstrate the benefit of the short-term and long-term temporal consistency provided by our method, with respectively the $\mathcal{L}_c$ term of the loss function defined in eq.(4.6) and the temporal linkage described in Subsection 4.2.4. Our method is able to recover the moving object segment when the object is temporarily static, showing its ability to segment amodally without any dedicated training, as shown in the first group of Fig.3.9. The second group highlights how our method can maintain the same mask labels in the video.

Results on Davis2017-motion

In addition to the datasets (DAVIS2016, FBMS59, SegTrackV2), we have evaluated our method on the DAVIS2017-motion dataset. DAVIS2017 [118] is an extension of DAVIS2016 dataset that includes additional videos with multi-object contents, resulting in multiple-segment annotations for the ground truth. It contains a total of 90 videos, split into 60 for training and 30 for validation. DAVIS2017-motion is a curated version of the DAVIS2017 dataset performed by the authors of [26] for a fair evaluation of motion segmentation based on flow information only, where connected objects sharing common motion are merged in the ground truth of the validation test.

Method / Scores	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
Ours	42.0	38.8	45.2
MoSeg	35.8	38.4	33.2
OCLR	55.1	54.5	55.7

Table 3.3 – Comparative evaluation on the DAVIS2017-motion validation set. The Jaccard index $\mathcal{J}$ expresses the correct overlap (intersection over union) between the extracted segments and the ground truth, while $\mathcal{F}$ focuses on segment boundary accuracy (the higher the better). $\mathcal{J}\&\mathcal{F}$ is the mean of the two. Evaluation is performed on the video as a whole, and reported scores are the average of the individual video scores.

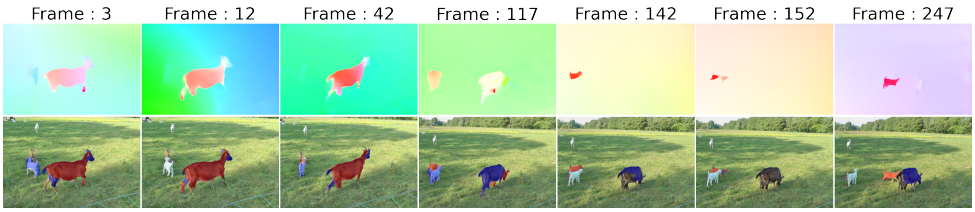


Figure 3.10 – Different instants of *goats1* video of FBMS59. First row: input optical flow (HSV color code). Second row: motion segments predicted by the network (before applying the global temporal linkage), and superimposed on the video frame (except background mask); when static, goats are merged with backgound.

We evaluated our method on the validation set using the official DAVIS-2017 evaluation algorithm that involves a Hungarian matching process over the sequence. Results are collected in Table 3.3. On this dataset, our method has a better $\mathcal{J}\&\mathcal{F}$ score than MoSeg [46] (42.0 *vs* 35.8), while OCLR flow-only [26] outperforms both (55.1), but OCLR is trained using synthetic data whose generation involves human annotation.

Differentiation of motion patterns

Network output may be limited to three segments with $K = 4$, not due to the model itself but to the train set. Indeed, DAVIS2016 train set comprises too few videos with several moving objects. We have noticed in other applications that the network, trained on a dataset involving many moving objects, is producing a number of segments equal to the specified K. Besides, we are not dealing with instance segmentation, but with motion segmentation into layers. Accordingly, objects with the same motion are prone to belong to the same mask (layer), but the decomposition into connected segments could be an easy postprocessing. On the other hand, our method manages to separate objects with different motions, e.g., two cars or two animals in Fig.4.7. In addition, results (with $K = 4$) on the *goats1* video of FBMS59 dataset are reported in Fig.3.10.

3.6 Partial Conclusion

We have designed an original unsupervised method [6] for the segmentation of multiple motions in a video. It fully leverages the temporal dimension of the motion segmentation problem. To the best of our knowledge, our method is the first unsupervised network-based OFS method involving short- and long-term temporal consistency, which leads to

6. https://github.com/Etienne-Meunier-Inria/ST-Space-Time-Flow-Segmentation

stable OF segmentation along the video. It introduces at training time spatio-temporal parametric motion models in sub-volumes, and a loss term expressing temporal consistency over consecutive masks while taking care of occlusions. In addition, the method allows for an easy temporal linkage of the motion segments throughout the video.

Our 3D network is flexible by design. It can straightforwardly handle different choices of mask number for the multiple motion segmentation. Different flow sub-volumes can be envisaged as input, including forward and backward in time. Besides, we have proposed an efficient way to code U-nets, which can be easily generalized beyond motion segmentation. Experimental results on several datasets demonstrate its efficiency and its accuracy by providing competitive results.

Nevertheless, our temporally-consistent motion segmentation method, if efficient for relatively short input sequences, may be limited for longer input sequences. Indeed, the linear temporal term of the spatio-temporal polynomial motion model may be too simple for longer time intervals likely to involve a different temporal evolution, and other network models as Transformers can handle more distant interactions. These extensions will be investigated in the next chapter.

Chapter 4

Long-Term Motion Segmentation

Human beings have the ability to continuously analyze a video and immediately extract the main motion components. Computer vision methods usually proceed frame by frame. We want to go beyond this classical paradigm and directly segment a video sequence as a whole. In the preceding chapter (Chapter 3), we have investigated a short-term temporally-consistent motion segmentation approach. In this chapter, we extend it to a long-term perspective.

As stressed in Chapter 3, the motion segmentation problem has a strong temporal dimension, as motion is generally consistent throughout the video (or within each video shot for long videos). The optical flow field at time t may be sufficient to get the segmentation at frame t of the video, leading to a motion-based video segmentation frame by frame. However, extending the time window can be beneficial. Beyond that, considering the video clip as a whole and directly providing the segmentation for the overall video clip is more in line with human vision, and should ensure a more reliable and efficient process.

4.1 Main Characteristics of the Proposed Method

We propose a novel long-term spatio-temporal model operating in a totally unsupervised way. We use the consecutive optical flow (OF) fields as input to decompose a video sequence into segments of coherent motion. More specifically, we have defined a transformer-based network for multiple motion segmentation that delivers temporally-coherent segmentation maps over the video sequence. It is trained in a completely unsupervised manner, without any manual annotation or ground truth data of any kind. The loss function combines a flow reconstruction term involving spatio-temporal parametric motion models, and a regularization term enforcing temporal consistency between successive masks. We model with B-splines the long-term temporal evolution of the motion parameters, which brings more flexibility than the linear temporal term (first-order

polynomial) of the motion model defined for the ST-MS method.

The use of a transformer decoder on the latent space of the 3D Unet allows interactions between inputs separated by arbitrarily temporal distances compared to the simple 3D convolutions that have a fixed receptive field. This is a key aspect of our design as we want to be able to segment long video sequences at test time. The impact of this choice is shown in Table 7.12. Furthermore, the decision to use only a transformer decoder, rather than a combination of encoder and decoder, is also important since it allows us to keep the computational cost and inference time reasonably low. Finally, on the conceptual side, the transformer decoder produces queries that represent each segment along with the segmentation, which is interesting because we might want to use them for downstream tasks (e.g., classification, motion clustering).

Our method also involves a latent represention of the segment motion augmented with positional embedding. In addition, the temporal linkage of the consecutive motion segmentations over a video sequence was designed as a relevant but *ad hoc* post-processing of the ST-MS method. Now, this post-processing is removed, and all is integrated in a long-term model that is end-to-end trained in a unsupervisedly way.

We will report experiments on four VOS benchmarks (DAVIS2016, DAVIS2017-motion, SegTrackV2, FBMS59). We also highlight through visual results the key contributions on temporal consistency brought by our method. In the following, we will call it LT-MS method.

4.2 Long-term Motion Segmentation Method

Recent years have seen the advent of transformer-based networks involving attention mechanisms [119]. Let us recall that, in [46], the authors used a transformer module, more specifically, the slot attention mechanism introduced in [120]. Divided attention is promoted in [49]. The resulting DivA method is based on the same principle as in [47] that motion segments are mutually uninformative. However, it is not limited to binary segmentation (dominant moving object *vs* background), but can segment a variable number of moving objects taking one optical flow field as input, and without the number of moving objects being specified by the user. It leverages a slot attention mechanism guided by the image content through a cross-modal conditional slot decoder.

We have designed a transformer-based network for multiple motion segmentation from optical flow. It is inspired from the MaskFormer architecture [13], but it only comprises

one head corresponding to the mask prediction, as described in Fig.4.1. The network takes as input a volume, of flexible temporal length, comprising several consecutive optical flow fields. Temporal consistency is expressed in two main ways at the training stage. Firstly, we associate a space-time motion model with each segment to characterize its motion along with the evolution of the latter over time. Secondly, we define a regularization term in the loss function enforcing stable labeling of the motion segments over the volume.

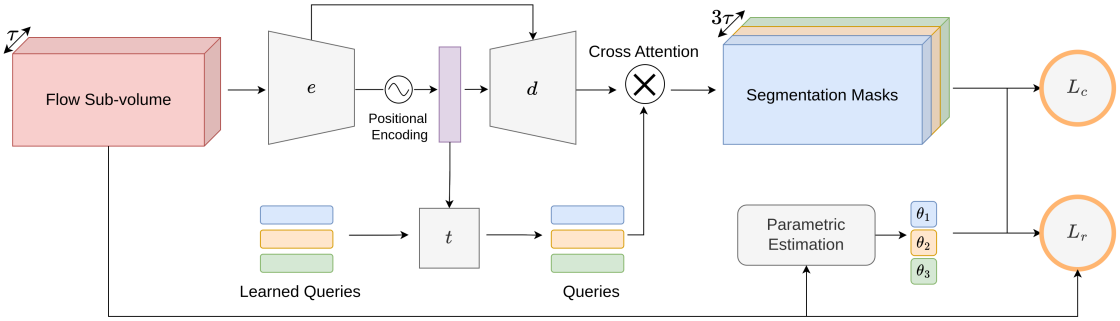


Figure 4.1 – Overall architecture of our multiple motion segmentation method ensuring temporal consistency with the consistency loss term $\mathcal{L}_c$ and the B-splines space-time motion models θ_k (for $k = 1, .., K$). It takes as input a volume composed of τ flow fields. It comprises a 3D U-net (e and d boxes) and a transformer decoder (t box). It also involves positional encoding. A cross-attention product yields the K segmentation masks corresponding to the input volume. For the sake of clarity, the block diagram is represented for three motion segments ($K = 3$). $\mathcal{L}_r$ is the flow-reconstruction loss term.

4.2.1 Spatio-temporal parametric motion model

We consider the (x, y, t)-space formed by a video. The set of τ consecutive frames will be designated as a space-time volume (or volume, to make it short). For short videos, the volume could be the whole video sequence. Our space-time motion model is defined by B-spline functions [121]. We have K motion segments. We assign a spatio-temporal parametric motion model $\tilde{f}_{\theta_k}$ to each motion segment $k, k = 1, .., K$. θ_k specifies the motion model $\tilde{f}$ for segment k. The motion model involves J parameters and each parameter $\theta_{k_j}, j = 1, .., J$, of the model results from a B-spline function of order n in the variable t over the space-time volume. In practice, we take $n = 3$. The number L of control points is given by $L = 2 + \lfloor \frac{\tau-2}{\nu} \rfloor$, where ν allows us to set the temporal frequency of control points. We put a control point at both ends of the volume (at $t = 1$ and $t = \tau$), and the other control points are equidistantly placed between the two. Most often, the

control points are not located at time points of the video sequence.

The space-time spline-based motion model is illustrated in Fig.4.2. Just for this illustration, the motion models are computed within the two segments, foreground moving object and background, provided by the ground truth. The estimated motion model for the foreground moving object is able to capture the periodic nature of the swing motion as demonstrated by the plots of the computed motion model parameters. Also, the motion model computed in the background segment perfectly fits the camera motion. Let us note that the articulated motion, specifically the woman's legs, should require multiple-motion segmentation to be correctly handled.

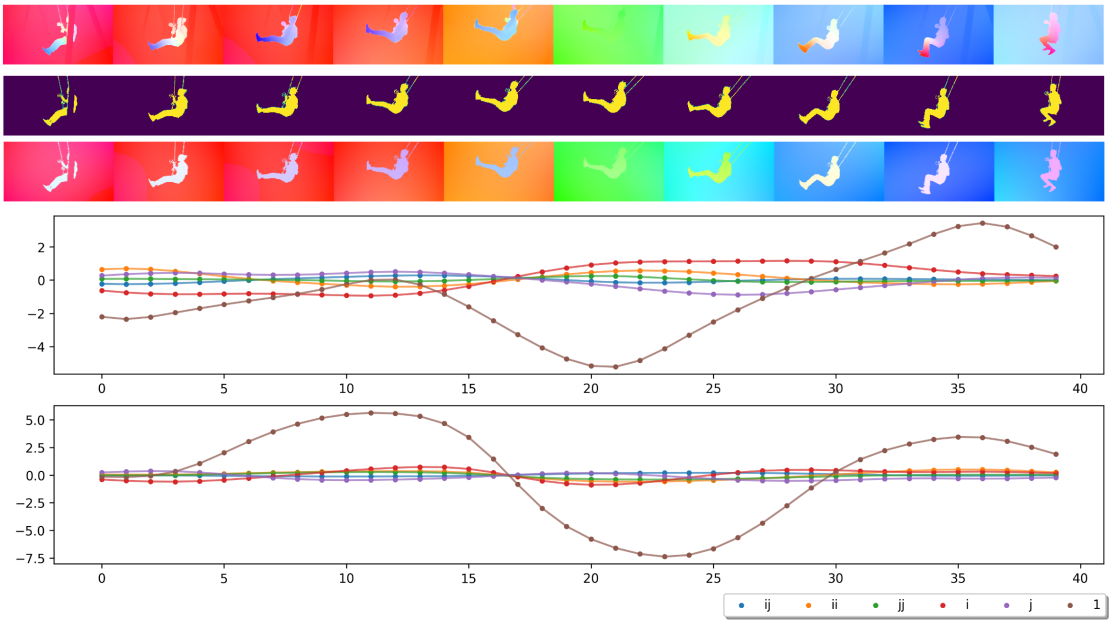


Figure 4.2 – Illustration of the space-time spline-based motion model. Top row: the input flows displayed with the HSV code for the *swing* video of DAVIS2016 dataset, the binary segmentation ground truth, the flows generated by the estimated spline-based motion models for the two segments. Middle-row: the plot of the temporal evolution of the six estimated model parameters corresponding to the flow u-coordinate for the foreground moving object. Bottom-row: the plot of the temporal evolution of the six estimated model parameters corresponding to the flow v-coordinate for the foreground moving object.

Any parametric motion model could be considered. We use the 12-parameter quadratic motion model to be able to account for continuously varying depth surface of the objects in the scene, especially for the whole background, and for complex object or camera motions. In contrast, the affine and the 8-parameter quadratic motion models assume

a planar object surface. Indeed, the latter exactly corresponds to the projection in the image of the 3D rigid motion of a planar surface. It is equivalent for velocity fields to the homography between planes. However, in presence of severe depth effects (strong depth discontinuities) and camera motion, the backgound cannot be represented by a single motion segment due to motion parallax corresponding to static objects located in the foreground.

The 2D flow vector yielded by the full quadratic motion model at any point (x, y) is given by:

$$\begin{aligned}\tilde{f}_\theta(x,y) = (&\theta_1 + \theta_2 x + \theta_3 y + \theta_7 x^2 + \theta_8 xy + \theta_9 y^2,\\ &\theta_4 + \theta_5 x + \theta_6 y + \theta_{10} x^2 + \theta_{11} xy + \theta_{12} y^2)^T.\end{aligned} \tag{4.1}$$

By arranging the motion model parameters as a 2x6 matrix:

$$\theta = \begin{pmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_7 & \theta_8 & \theta_9 \\ \theta_4 & \theta_5 & \theta_6 & \theta_{10} & \theta_{11} & \theta_{12} \end{pmatrix}, \tag{4.2}$$

we can rewrite eq.(4.1) in a more compact way as a matrix product:

$$\tilde{f}_\theta(i) = \theta\ c(i), \tag{4.3}$$

with $c(i) = (1, x, y, x^2, xy, y^2)^T$ being the position expansion for site $i = (x, y)$.

4.2.2 Loss function

The loss function of our motion segmentation network is composed of two terms as in the ST-MS method: a segment-wise flow reconstruction term and a temporal consistency one on the predictions. The first term, denoted $\mathcal{L}_r$, expresses how the estimated parametric motion models fit the input optical flow within each segment. It writes:

$$\mathcal{L}_r = \frac{1}{\tau} \sum_{s=1}^{\tau} \frac{\sum_{i \in \mathcal{I}} \sum_{k=1}^{K} m_k(i, t+s) \| f(i, t+s) - \tilde{f}_{S_n(\theta_k)}(i, t+s) \|_1}{\sum_{i \in \mathcal{I}} \| f(i, t+s) \|_1}, \tag{4.4}$$

where i is a site of the image grid $\mathcal{I}$, K is the number of motion layers or segments, $f(i, t)$ is the flow vector at site i and time instant t, and $m_k(i, t)$ denotes the probability of site i to belong to motion segment k at time t, that is, the prediction (or output)

of the motion segmentation network. f_t will designate the optical flow field at time t, $f_t = \{f(i,t), i \in \mathcal{I}\}$. We normalize by the norm of the input flows to hinder the impact of the flow magnitude. By denoting $S_n(\theta_k)$ the subscript of $\tilde{f}$, we want to emphasize that the motion model parameters for each segment k are estimated through the B-spline functions. More specifically, we have:

$$\tilde{f}_{S_n(\theta_k)}(i,t) = \sum_{l=1}^{L} \tilde{f}_{\theta_{k,l}}(i,t) B_{n,l}(t), \tag{4.5}$$

where $B_{n,l}$ is the l^{th} B-spline and $\theta_{k,l}$ corresponds to the l^{th} control point of the spline function.

We use the robust norm L_1 in eq.(4.4) to overcome the presence of outliers in the motion segment, especially at the beginning of the training when segments are not yet well extracted, and to mitigate possibly wrong flow vectors, around motion discontinuities in particular.

The second term, denoted $\mathcal{L}_c$, enforces temporal consistency of the motion segments. To do this, the probability of site i to belong to segment k is assumed to be stable over time. We have:

$$\mathcal{L}_c = \frac{1}{2K|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{k=1}^{K} \sum_{t=1}^{\tau-1} |m_k(i,t) - m_k(i,t+1)|, \tag{4.6}$$

where $|\mathcal{I}|$ designates the number of sites over the image grid $\mathcal{I}$. For the sake of simplicity, we have adopted an Eulerian standpoint [117], that is, we compare segment labels over time at any given site i of the image grid. In fact, every point is likely to move and a Lagrangian standpoint [117] would be more appropriate. It would require the use of the optical flow vectors to track every point over time. However, the computed flow field may be imprecise or even erroneous at some points, and besides, interpolation operations would be necessary since the flow components take on real values. The Eulerian standpoint works on the overlap of the successive positions of the moving parts, but not on occluded or disoccluded parts. This justifies the use of the L_1 norm to deal with the latter configuration.

As in the ST-MS method, we further prevent from enforcing the temporal consistency over occlusion areas by ignoring sites i in the summation over $\mathcal{I}$ in rel.4.6 that exhibit a large temporal flow difference. More precisely, we set a threshold λ so that a quantile η

of sites i is discarded as follows:

$$p(\|f(i,t+\tau)-f(i,t)\|_1 \geq \lambda) \leq \eta. \tag{4.7}$$

In practice, we take $\eta = 1\%$. In doing so, we make an implicit assumption on the overall surface of the occlusion areas, but it seems reasonable for the datasets we deal with. The loss function is the sum of the two terms:

$$\mathcal{L} = \mathcal{L}_r + \beta\mathcal{L}_c. \tag{4.8}$$

We simply set $\beta = 1$, since the two terms of the loss function are properly normalized.

Let us stress that we impose both short-term and long-term temporal consistency. The loss term $\mathcal{L}_c$ (4.6) primarily expresses a short-term consistency effect by constraining the mask probability $m_k(i,t)$ at a given site i between two consecutive time instants t and $t+1$. Nevertheless, it also induces a long-term effect by propagation from near to near. The motion model leads to long-term consistency effect by construction, since it is defined over the volume with the spline function. However, it also involves a short-term effect around the control points.

4.2.3 Network architecture

The overall architecture of our unsupervised multiple motion segmentation framework is illustrated in Fig.4.1. It includes two main modules. The first one, taking the flow subvolume as input, is a 3D U-net [86]. The latent code augmented with embedding positions is directed to the transformer decoder. Then, by cross-attention, the output formed by the volume of segmentation masks is produced. The training of the overall architecture is based on the minimization of the loss function defined below, while the motion model parameters of the segments are given by the estimation of the B-spline functions. Temporal consistency is provided both by the loss function and the space-time motion models.

4.2.4 Segment selection for evaluation

As we already mentioned in the two preceding chapters, to evaluate our method and compare it with similar unsupervised methods, we use VOS benchmarks as a substitute for optical flow segmentation (OFS) benchmarks, since no such benchmarks are available.

The two tasks are close. However, the VOS one is attached to the notion of a primary object of interest moving in the foreground (sometimes, a couple of objects). As a consequence, we have to select the right segments to cope with the binary ground truth of the VOS benchmarks, as usually done for the DAVIS2016, SegTrackV2 and FBMS59 datsets.

Since we deal with multiple-motion segmentation, i.e., K segments, we have to group them into two clusters corresponding to the foreground moving object on one side and the background on the other side. We proceed as we did in Chapter 3 for our ST-MS method (details given in Annex 7.2, eq.(3.11). However, this time, we can directly evaluate our LT-MS method in one shot over the predicted segmentation sequence, since there is no temporal linking postprocessing for LT-MS.

Regarding the evaluation on the DAVIS2017-motion dataset whose ground truth is not binary, we also achieved it as we did for our ST-MS metod. We use the Hungarian matching method to associate each predicted segment k with the ground-truth annotations, knowing that we have to consider $K = 3$ masks for this experimentation.

4.3 Implementation

4.3.1 Implementation details

Following [3], [21], [46], [55], we adopt the RAFT method [56] to compute the optical flow fields. More specifically, we use on our side the RAFT version trained on the MPI Sintel dataset [97]. We downsample the computed flow fields to provide the network with 128×224 vector fields as input. The output segmentation maps are consequently upsampled to the original frame size for evaluation w.r.t. the ground truth. Thus, we achieve much more efficient training and inference stages. We typically use subvolumes of temporal length $\tau = 9$ in the training stage, but, at test time, we can process subvolumes of longer length.

Regarding the estimation of the spatio-temporal parametric motion model, since the x and y coordinates are normalized within $[-1, 1]$, we apply a similar normalization for the t coordinate. In all the reported experiments, we use the full quadratic motion model with 12 parameters, and we set $\nu = 3$ for the frequency factor in the B-spline function.

We use the prediction $m_k(i, t)$ of the network to decide to which segment k site i belongs to at time instant t. More precisely, we select for each site i the segment $\hat{k}$ with the highest probability.

Our LT-MS method is very efficient at test time. The computational time amounts on average to 210 *fps* on a GPU P100, that is, twice as fast as the preceding ST-MS method. It is certainly due to the long flow sequence given as input to the network, which allows for parallelisation of some heavy computations. In addition, our LT-MS architecture remains light since it combines only three Unet layers and a transformer decoder on the downsampled feature space.

4.3.2 Data augmentation and network training

We perform two types of data augmentation. The first one consists in adding a global flow to the input flow as we did previously for the EM and ST-MS methods. The global flow is now given by a full spline-based spatio-temporal motion model whose parameters are chosen at random. We just make sure that the added global flow is roughly equivalent in magnitude to the initial flow. The same global flow is added to the flow fields of the input volume. This type of data augmentation allows to mimic different camera motions, enforcing that the motion segments are independent of it. For the second type of data augmentation, we corrupt a few input flow out of the nine ones. The idea is to simulate a poorly estimated flow field and to compel the temporal consistency to compensate.

Our motion segmentation method is entirely unsupervised. We do not perform any manual annotation in all the experiments. We train the overall motion segmentation network on the FlyingThings3D (FT3D) dataset [100] once for all. This will ensure that our network generalizes well to any unseen datasets. Moreover, the stopping epoch is selected from the loss function evaluated on the DAVIS2016 training set.

We use Adam optimizer with the following strategy on the learning rate to train the network (3D Unet and transformer). We inspired from the warmup-decay strategy in [26] for the learning rate. We linearly increase it from 0 to $1e-4$ for 20 epochs then divide it by two every 40 epochs.

The estimation of the motion model parameters through the B-spline function is achieved with the Pytorch implementation of L-BGFS [103]. Let us stress again that we estimate the parametric motion models only at training time.

4.4 Experimental Results

4.4.1 Datasets

We have carried out experiments again on the four datasets: DAVIS2016 [1] [15], SegTrackV2 [2] [88], FBMS59 [51], and DAVIS2017-motion [26].

Let us recall that DAVIS2016 consists of 50 videos (and 3455 frames) that are split in a training set of 30 videos and a validation set of 20 videos. They contain diverse moving objects. Only the primary moving object is annotated in the ground truth. The criteria for evaluation on this dataset are the Jaccard score (denoted by $\mathcal{J}$), i.e., intersection-over-union, and the contour accuracy score (denoted by $\mathcal{F}$).

SegTrackV2 includes 14 videos (with a total of 1066 annotated frames), and FBMS59 contains 59 videos (720 annotated frames), both involving one moving object but sometimes a couple of moving objects. For FBMS59, we use the 30 sequences of the validation set. Although annotations may comprise multiple objects, the community usually exploits SegTrackV2 and FBMS59 in the same way as DAVIS2016, i.e., benchmarks with a binary ground-truth, by grouping objects in the foreground. We will follow this practice for SegTrackV2 and FBMS59.

As explained in Chapter 3, DAVIS2017 [118] contains a total of 90 videos, split into 60 for training and 30 for evaluation. Let us recall that DAVIS2017-motion is a curated version of the DAVIS2017 dataset performed by the authors of [26] for a fair evaluation of motion segmentation based on flow information only. Connected objects sharing common motion are merged in the ground truth of the validation test. We evaluate our method on the validation set as in [26], using the official evaluation algorithm that involves a Hungarian matching process.

4.4.2 Ablation study

We have conducted an ablation study to assess three main components of our method LT-MS with four masks ($K = 4$), in particular related to the temporal dimension of the problem. We have changed one component at a time as specified below:

— use of the polynomial space-time quadratic motion model of ST-MS instead of the space-time motion model based on B-splines over the input sequence,

1. https://davischallenge.org/index.html
2. https://paperswithcode.com/dataset/segtrack-v2-1

Ablation / Dataset	DAVIS2016		FBMS59		SegTrackv2	
Cut Size	10	120	10	120	10	120
Full Model LT-MS-K4	74.8	72.4	61.0	58.2	61.3	60.4
Unet3D only	73.0	71.3	56.6	55.5	58.2	57.3
No consistency term $\mathcal{L}_c$	73.5	71.0	57.5	55.5	58.0	57.5
Polynomial space-time quadratic model	73.4	69.8	57.4	54.5	57.8	56.6

Table 4.1 – Ablation study for three main components of our method LT-MS ($K = 4$) on the three datasets DAVIS2016, FBMS59 and SegTrackV2. Only one model component is modified at a time. The performance scores are given by the Jaccard index $\mathcal{J}$; the higher $\mathcal{J}$, the better. We report ablation results with two input-flow sequence length (or cut size), respectively, by dividing the video into pieces of ten successive frames or by considering 120 successive frames (in practice, the whole video for the DAVIS2016 dataset).

— loss function without the consistency term $\mathcal{L}_c$,

— just the convnet without the transformer decoder.

All the ablation experiments were run on the three datasets, DAVIS2016, FBMS59, SegTrackV2. Results are collected in Table 7.12. In addition, we performed them for two input sequence configurations, respectively input sequences of ten flows, and input sequence of 120 flows (in practice, the whole video for the DAVIS2016 dataset).

We can observe that the three ablations have almost the same impact on the performance. The three corresponding model components, i.e., the spline-based motion model, the temporal-consistency loss term, and the transformer decoder are thus all beneficial in similar proportions. They are able to handle the temporal dimension of the problem and the temporal motion evolution along the sequence in a compelling way. In addition, Fig.4.3 contains the performance scores for the three ablations when the length of the input flow sequence varies from 10 to 120. Overall, they exhibit comparable behaviour at a certain distance from the full model.

Admittedly, the contributions of these three components are more significant for the FBMS59 and the SegTrackV2 datasets. However, the dynamic content of the majority of the DAVIS2016 videos, and then, the overall performance score, cannot allow us to fully appreciate the contributions of these three model components. Yet, they can be acknowledged by visualizing results obtained on some videos of DAVIS2016, as shown in Fig.4.4 and Fig.4.5.

Visual results reported in Fig.4.4 clearly demonstrate that the addition of the temporal consistency loss term $\mathcal{L}_c$ allows us to get far more consistent segments over time, whether

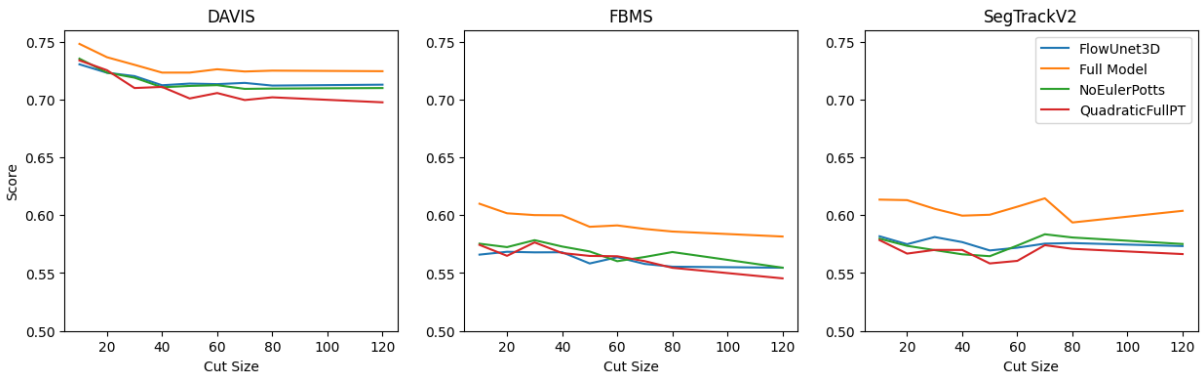


Figure 4.3 – Influence of the length of the input flow sequence (also referred to as cut size on the horizontal axis of the plot) on the three modified versions of our LT-MS-K4 method with comparison with the full model. Left plot: for the DAVIS2016 dataset. Middle plot: for the FBMS59 dataset. Right plot: for the SegTrackV2 dataset. Overall, they exhibit comparable behaviour at a certain distance from the full model.

for the background, or the moving objects. For instance, in the fourth example, the foreground moving car is perfectly segmented throughout the sequence along with its wheels, and (small) moving cars in the background. Fig.4.5 highlights the contribution of the spline-based motion model on the *dog* video, and its obvious ability to handle motions that do not vary uniformly, as the erratic movement of the dog.

4.4.3 Quantitative and comparative evaluation

We report in Table 4.2 the results obtained by two versions of our LT-MS method on the three datasets DAVIS2016, SegTrackV2, and FBMS59. LT-MS-K2 performs segmentation with only two masks ($K = 2$) as our EM method, while LT-MS-K4 involves four masks ($K = 4$) as done for our ST-MS method. Table 4.2 also collects results obtained by our two preceding methods, EM and ST-MS, and other existing methods when available. Since our method is fully unsupervised and only uses optical flow as input, our comments will focus on similar methods for a fair comparison. We follow the categorization we initially proposed in Chapter 2 or in [55], regarding input and training, by the way very close to other propositions. However, we have added a category w.r.t. the network input for four recent methods, [3], [5], [21], [106], that only use RGB images as imput at test time, the optical flow being only involved in the loss function. Evaluation is performed on the binary ground truth (foreground moving object *vs* background) for the three datasets.

We still put the OCLR method [26] in the category of unsupervised methods, whereas

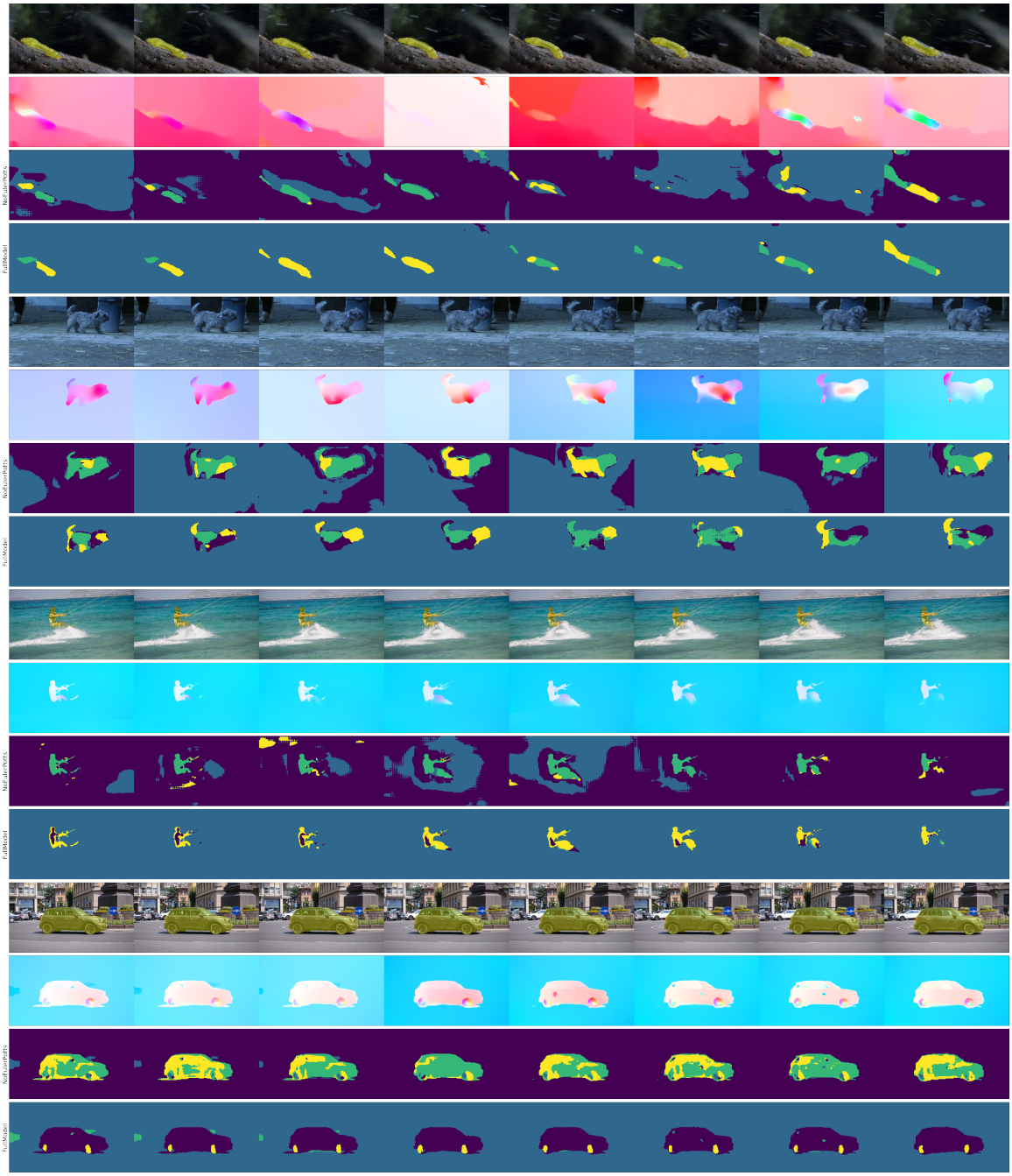


Figure 4.4 – Four groups of qualitative results regarding the ablation of the temporal-consistency loss term. They respectively correspond to the *worm* video of SegTrackV2, the *dogs01* video of FBMS59, and the *kite-surf* and *car-roundabout* videos of DAVIS2016. For each group, the first row contains sample images with the segmentation ground-truth, when available at that frame, overlaid in yellow, the second row displays the input flows, the third and fourth rows show the predicted motion segmentations, respectively without and with the temporal consistency loss term. Clearly, this model component allows us to get far more consistent segments over time.

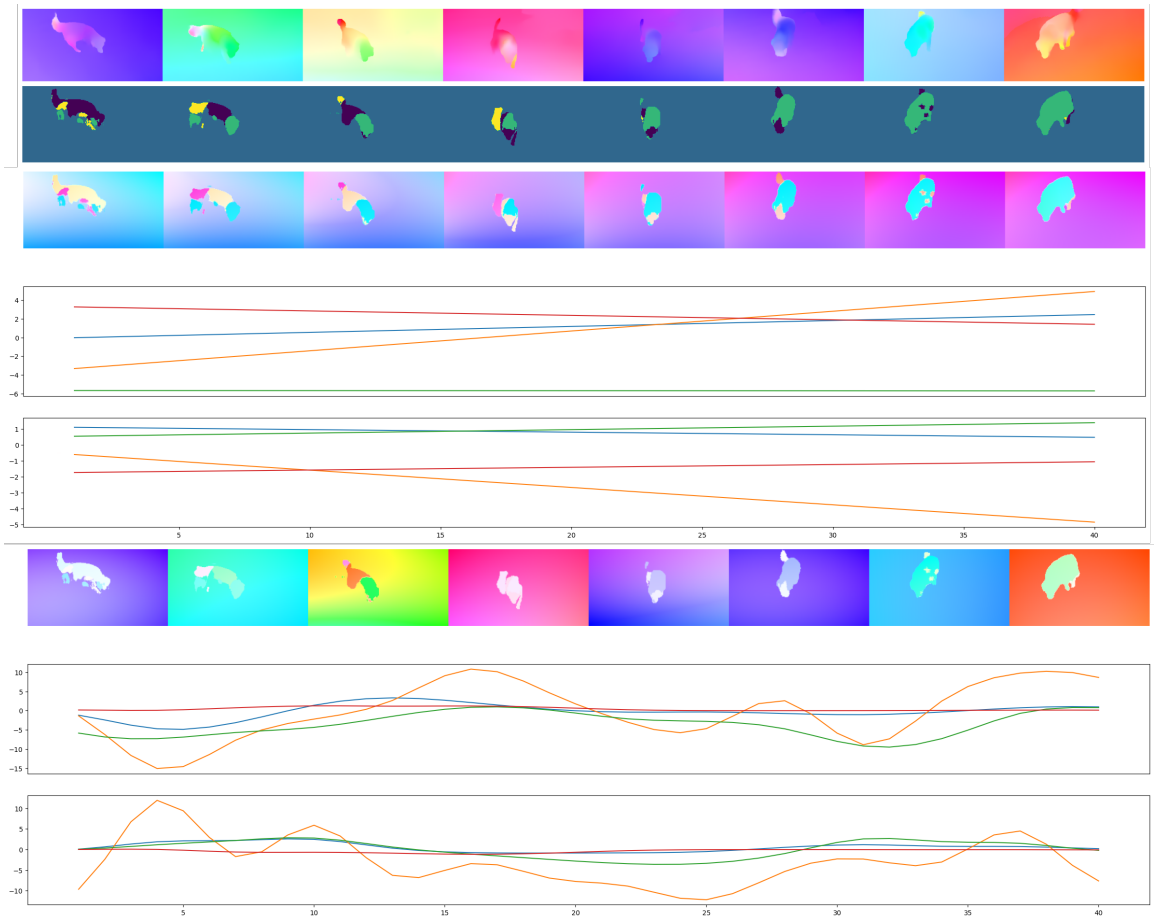


Figure 4.5 – Visual assessment of the contribution of the spline-based motion model on the *dog* video of DAVIS2016 dataset (with $K = 4$). From top to bottom: input flows of the sequence; corresponding predicted motion segmentation maps; flows reconstructed with the space-time polynomial motion model; plots over time of the mean-value of the u-components of the flows provided per segment by the space-time polynomial motion models; plots over time of the mean-value of the v-components of the flows provided per segment by the space-time polynomial motion models; flows reconstructed with the space-time spline-based motion model; plots over time of the mean-value of the u-components of the flows provided per segment by the spline-based motion models; plots over time of the mean-value of the v-components of the flows provided per segment by the spline-based motion models. Clearly, the space-time polynomial model fails to handle non-uniformly varying motion, whereas the spline-based model does.

Method	Training	Input	DAVIS2016 $\mathcal{J}\uparrow$	DAVIS2016 $\mathcal{F}\uparrow$	SegTrack V2 $\mathcal{J}\uparrow$	FBMS59 $\mathcal{J}\uparrow$
LT-MS-K4	Unsupervised	Flow	74.8	72.2	61.3	61.0
LT-MS-K2			70.3	68.5	58.6	55.3
ST-MS (Chapter 3) [122]			73.2	70.3	55.0	59.4
EM (Chapter 2) [55]			69.3	70.7	55.5	57.8
MoSeg [46]			68.3	66.1	58.6	53.1
FTS [104]			55.8	-	47.8	47.7
TIS_0 [68]			56.2	45.6	-	-
OCLR* [26] (flow only)			72.1	-	67.6	65.4
GWM [3]		RGB (Flow in loss)	79.5	-	78.3	77.4
RCF [21]			80.9	-	76.7	69.9
AMD [5]			57.8	-	57.0	47.5
MOD [106]			73.9	-	62.2	61.3
DivA(4)† [49]		RGB & Flow	72.4	-	64.6	-
TIS_s [68]			62.6	59.6	-	-
CIS - No Post [47]			59.2	-	45.6	36.8
CIS - With Post [47]			71.5	-	62.0	63.6
DyStab - Dyn [48]	Supervised Features	Flow	62.4	-	40.0	49.1
DyStab - Stat&Dyn [48]		RGB & Flow	80.0	-	73.2	74.2
ARP [27]			76.2	70.6	57.2	59.8
MATNet [19]	Supervised	RGB & Flow	82.4	80.7	50.4	76.1
COSNet [24]		RGB	80.5	79.5	49.7	75.6

Table 4.2 – Results obtained with two versions of our LT-MS method on DAVIS2016, SegTrackV2, FBMS59. LT-MS-K2 performs segmentation with only two masks ($K = 2$) as our EM method, and LT-MS-K4 involves four masks ($K = 4$) as done for our ST-MS method. We also include comparison with our two preceding methods, EM and ST-MS, and unsupervised and supervised methods (scores from cited articles). All scores regarding DAVIS2016, SegTrackV2, and FBMS59 corresponds to the evaluation on the binary ground-truth. For LT-MS-K4 and LT-MS-K2, we report results obtained with a cut size of 10. The Jaccard index $\mathcal{J}$ expresses the correct overlap (intersection over union) between the extracted segments and the ground truth, while $\mathcal{F}$ focuses on segment boundary accuracy (the higher the better for both). $\mathcal{J}\&\mathcal{F}$ averages the two. Performance is assessed by the average score over all samples, for all datasets but DAVIS2016. For the latter, the overall score is given by the average of sequence scores. *Actually, OCLR is not a truly unsupervised method, and the authors of DivA do not include OCLR among the unsupervised methods. Indeed, it relies on human-annotated sprites to include realistic shapes of moving objects in the synthetic data used at training time. †DivA also uses images since its decoder network leverages input images (conditional decoder). The authors of [49] provide results on FBMS59 only for the multi-object setting.

the authors of the DivA method [49] did not. Indeed, OCLR is not fully unsupervised, since it relies on human-annotated sprites to include realistic shapes in the computer-generated data used at training time. We consider the OCLR version taking only optical flow as input. The post-processing possibly added to the CIS method [47], based on Conditional Random Fields (CRF), is an heavy one, which leads most authors to retain only the version without post-processing for a fair comparison.

Our LT-MS method (both LM-MS-K4 and LT-MS-K2) delivers very convincing results compared to unsupervised methods taking optical flow as input, especially for the DAVIS2016 dataset that is the main VOS benchmark. 0ur method also performs very well on the two other datasets. It is the second best for FBMS59, outperformed by OCLR. OCLR and DivA demonstrate better performance than LT-MS-K4 on the SegTrackV2 dataset. However, as aforementioned, OCLR is not a fully unsupervised method, while DivA leverages RGB images at some point. Indeed, in DivA, the decoder network leverages input images (conditional decoder). In addition, DivA, along with MoSeg and CIS methods, takes in turn four flow fields as input for a given time instant t, i.e., flows between t and $t+1$, t and $t+2$, t and $t-1$, t and $t-2$, and then, averages the four corresponding predictions to get the final result. Finally, SegTrackV2 includes sequences acquired with a poorly controlled handheld camera, which leads to unstable sequences where the contribution of our method is therefore less likely to be emphasized.

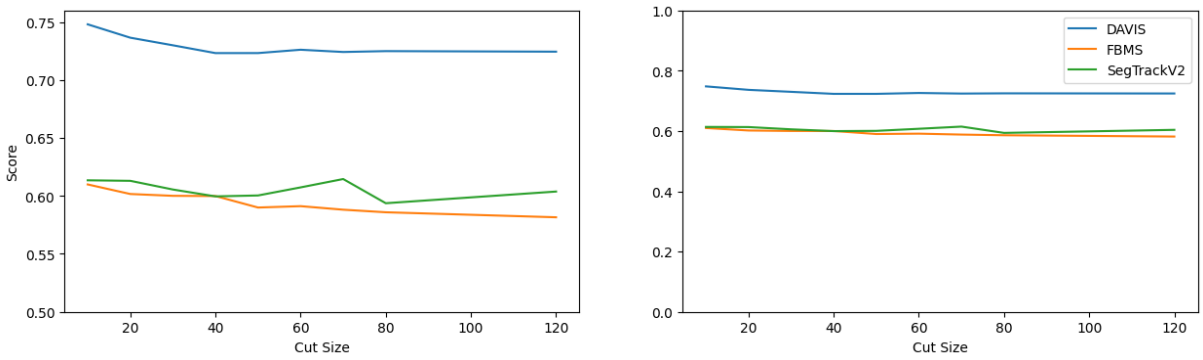


Figure 4.6 – Influence of the length of the input flow sequence (also referred to as cut size on the horizontal axis of the plot) on the LT-MS-K4 method for three datasets, DAVIS2016, FBMS59 and SegTrackV2. As expected, the best performance is obtained for the smallest length (cut size equal to 10), but performance decreases slowly when the cut size increases and remains stable for larger ones. On the right, the plot is shown with the full range of values on the ordinate $[0, 100]$. On the left, a zoomed-in version of the plot.

We have also tested how our LT-MS-K4 method behaves when varying the length of the input flow sequence. Results are plotted in Fig.4.6 for three datasets, DAVIS2016, FBMS59 and SegTrackV2. As expected, the best performance is obtained for the smallest length (equal to 10) of the input flow sequence. However, performance decreases slowly when the length increases, and remains stable for larger ones. It demonstrates the intrinsic ability of the LT-MS method to achieve accurate and consistent motion segmentation over long periods of the video, which is a unique property. We also did it for LT-MS-K2 as reported in Table 4.3. We even get slightly better results for DAVIS2016 and SegTrackV2 when the video sequence is processed in one go, i.e., with an infinite cutsize.

Dataset	DAVIS2016		FBMS59		SegTrackV2	
Cut Size	10	∞	10	∞	10	∞
LT-MS-K2	70.3	70.7	55.3	48.7	58.6	59.3

Table 4.3 – Results obtained on the three datasets DAVIS2016, FBMS59 and SegTrackV2 with LT-MS-K2 for respectively a cut size of 10 and no cut (the video sequence is processed in one go).

Method / Scores	$\mathcal{J}\&\mathcal{F}\uparrow$	$\mathcal{J}\uparrow$	$\mathcal{F}\uparrow$
LT-MS-K3	42.2	39.3	45.0
ST-MS (Chapter 3)	42.0	38.8	45.2
MoSeg [46]	35.8	38.4	33.2
OCLR* [26]	55.1	54.5	55.7

Table 4.4 – Comparative evaluation on the DAVIS2017-motion validation set for three masks ($K = 3$). The Jaccard index $\mathcal{J}$ expresses the correct overlap (intersection over union) between the extracted segments and the ground truth, while $\mathcal{F}$ focuses on segment boundary accuracy (the higher, the better). $\mathcal{J}\&\mathcal{F}$ is the mean of the two. Evaluation is performed on the video as a whole, and reported scores are the average of the individual video scores. *Actually, OCLR is not a truly unsupervised method, and the authors of DivA do not include OCLR among the unsupervised methods. Indeed, it relies on human-annotated sprites to include realistic shapes of moving objects in the synthetic data used at training time.

We have also performed the evaluation of multiple motion segmentation for a multi-segment setting. Since multiple-motion segmentation is harder than the binary motion segmentation (moving foreground *vs* background), accuracy scores are expected to decrease for all methods. In Table 4.4, we report comparative results on the DAVIS2017-motion dataset. As for the other methods, we performed segmentation with three masks.

Method / Scores	bIoU	Linear Assignment
DivA [49]	42.0	-
Ours K=3	58.3	47.2
Ours K=4	56.8	44.6

Table 4.5 – Results on the FBMS59 dataset for multi-object detection using two different evaluation metrics. Bootstrap IoU (bIoU) is a metric described in [49] where each ground truth mask is mapped to the most likely predicted segment. This mapping is performed at the frame level, and thus, this evaluation actually does not take into account temporal consistency. Linear assignment corresponds to a bilinear mapping between the ground-truth labels and the predicted segments at the sequence level. This metric is similar to the one used in the official DAVIS 2017 evaluation [118].

To this end, we finetuned the LT-MS-K4 network on the DAVIS2016 training set with now three masks ($K = 3$). The resulting performance is slightly better than ST-MS. In Table 4.5, we report multimask segmentation results for the FBMS59 dataset with two different metrics. Results with $K = 4$ are obtained with the same network LT-MS-K4 as the one delivering the results reported in Table 4.2. Results with LT-MS-K3 ($K = 3$) correspond to the model introduced in Table 4.4. Our LT-MS method outperforms the DivA method.

Overall, temporal consistency is properly handled over long temporal periods by our LT-MS-K4 method which delivers excellent segmentation performance. Beyond segmentation performance, we want to stress that our method is the only one providing *by design* a coherent segmentation over the sequence, which is a significant added value. Thus, we can claim that we have not only segmented the moving objects throughout the sequence, but also achieved some kind of tracking.

4.4.4 Qualitative visual evaluation

Fig.4.7 contains several visual results to demonstrate how our method behaves on different situations. We display six result samples obtained on different videos of the benchmarks. From top to bottom of Fig.4.7, we have the *motocross-jump* video from DAVIS2016, the *monkey* and *hummingbird* videos from SegTrackV2, the *people2* and *goats01* videos of FBMS59, and finally, the *libby* video from DAVIS2016.

We can observe that the segmentation are globally accurate. Let us note that the ground-truth is not necessarily available for all the video frames depending on the datasets. Since our method can involve several masks, we can properly handle articulated motions

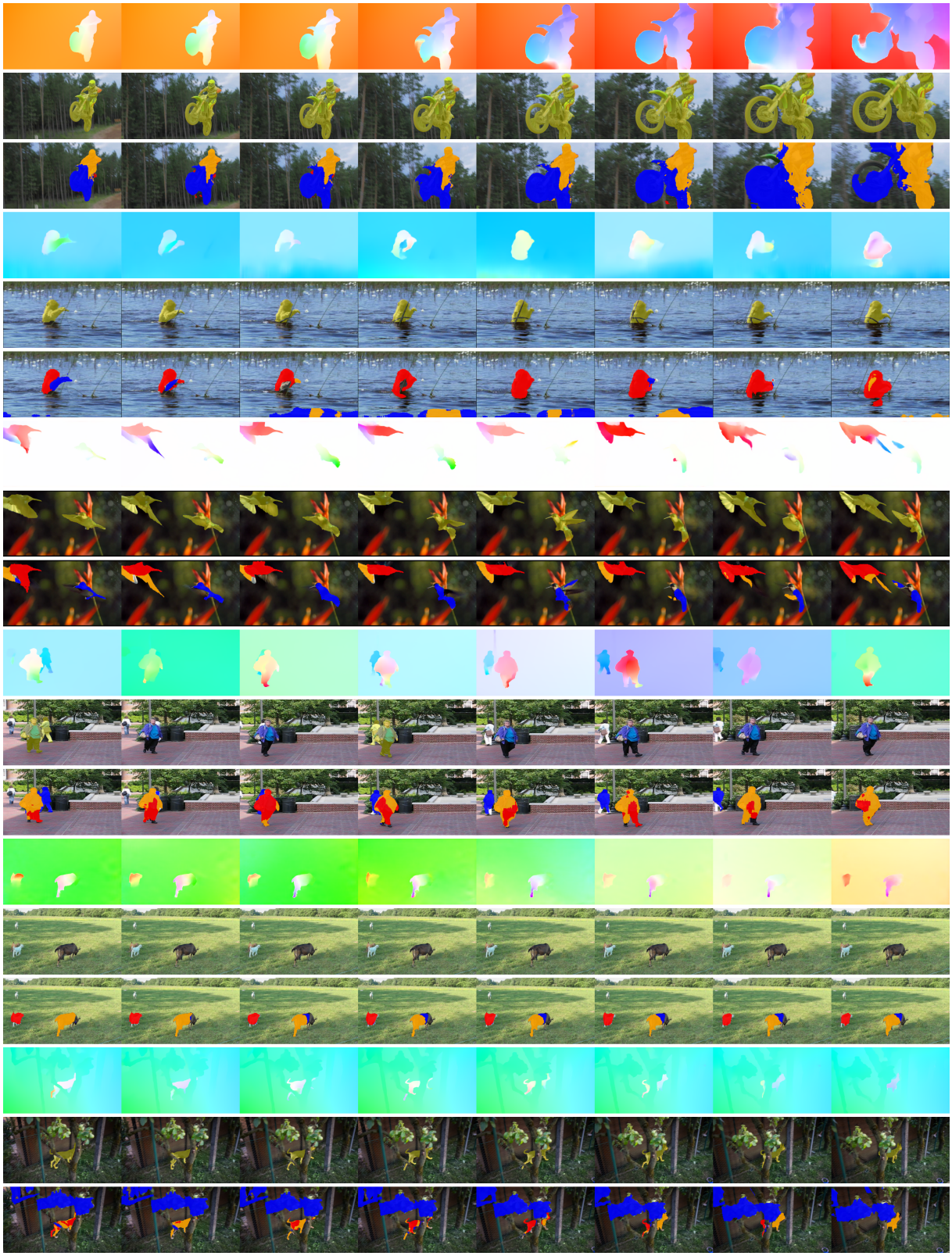


Figure 4.7 – Results obtained with our LT-MS-K4 method, i.e., using four masks ($K = 4$). Six groups of results are displayed. For each group, we have three rows. The first row samples successive flow fields corresponding to the processed video, the optical flow fields being represented using the usual HSV color code. The second row contains the corresponding images of the video, where the ground-truth segment corresponding to the moving object is overlaid in yellow when available at that frame. The third row shows the motion segments provided by our LT-MS-K4 method (given by layers that are not necessarily connected) with one colour per segment. For all the results, we adopt the same color set for the three masks corresponding to the moving objects (blue, red and orange), and we let the background image for the background mask for the sake of clarity. These examples are drawn from the different datasets.

(*hummingbird, monkey, people2*), deal with the presence of several moving objects in the scene (*goats01, hummingbird, people2*), separate the rider from the motorbike (*motocross-jump*), or accomodate motion parallax (*libby*). We must keep in mind that our actual target is the optical-flow segmentation (OFS) task, even if we evaluate our method on VOS benchmarks. Since the VOS benchmarks mainly deal with the segmentation of one primary object moving in the foreground, it may occur some discrepancies with OFS, which negatively impacts the evaluation scores. The segmentation of additional parts, which appears wrong w.r.t. VOS ground truth, on the contrary makes sense from the OFS standpoint.

In Fig.4.8, we collect additional visual results on three videos of the SegTrackV2 dataset, from top to bottom, the *birdfall*, *bird-of-paradise*, and *bmx* videos. Results demontrate the ability of our long-term segmentation method to recover, to some extent, correct segmentation even from flow fields wrongly computed at some time instants.

4.5 Partial Conclusion

We have designed an original transformer-based unsupervised method for the segmentation of multiple motions in a video. It fully acknowledges the temporal dimension of the motion segmentation problem. Indeed, to the best of our knowledge, our method is the first unsupervised network-based OFS method explicitly leading to a stable and consistent OF segmentation throughout long video sequences. It introduces at training time, on one hand, B-splines spatio-temporal parametric motion models over space-time volumes, and on the other hand, a loss term expressing temporal consistency over successive masks while taking care of occlusions. Experimental results on several datasets demonstrate its efficiency and its accuracy by providing competitive results. In addition, our LT-MS method is very fast at test time.

Our transformer-based network is flexible by design in several ways. At test time, it can be applied to input volumes of any time length, in any case longer than the one used at training time. It can accomodate different choices of mask number made by the user before training for multiple motion segmentation. However, changing the number of masks requires to retrain the model. More flexibility could be obtained by introducing a slot attention mechanism, which should allow us to modify the number of masks at test time without retraining the model. We report in Annex 7.3 an extension of the LT-MS model involving slot attention. However, results are not that convincing so far.

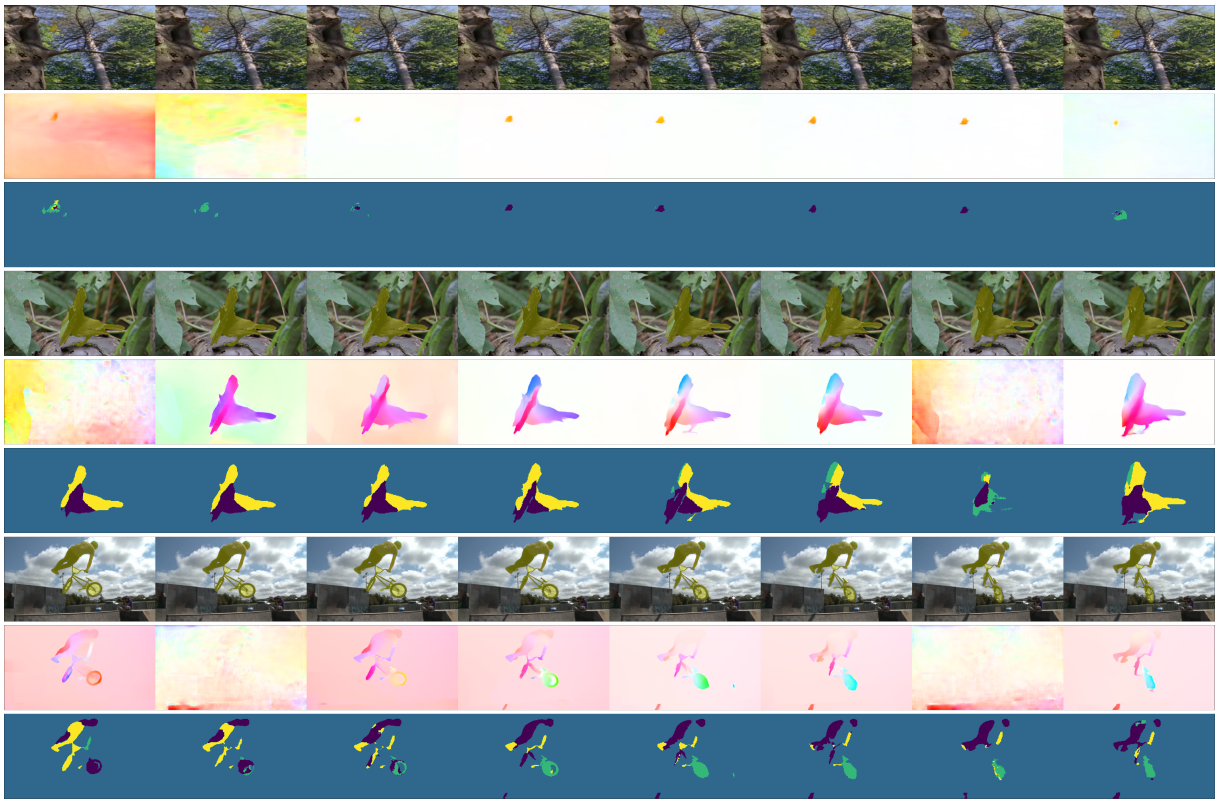


Figure 4.8 – Illustration of the temporal consistency provided by our LT-MS-K4 method for three examples, the *birdfall*, *bird-of-paradise*, and *bmx* videos of SegTrackV2. For each group, the first row contains the video images with the ground truth overlaid in yellow when available; the second row depicts the corresponding flow fields represented with the HSV code while normalized independently from each other; the third row provides the predicted segmentation.

Chapter 5

Gradient-based Interpretation of a Frame Classification Network for Motion Saliency

In this chapter, we propose a new paradigm for the unsupervised computation of motion saliency maps. We estimate these maps from the interpretation of a frame-based motion saliency classification network with optical flow (OF) as only input.

Motion saliency (MS) is an important issue in dynamic scene analysis. We formulate MS as a meta-task that can be instantiated for different tasks usually handled independently. To support this claim, we have addressed two important computer-vision problems with this MS paradigm: independent motion segmentation and anomalous motion detection in videos.

Our paradigm can accommodate a given form of motion saliency by simply training the frame-based classification network on the corresponding task. Moreover, our MS estimation is unsupervised, as it does not require any ground-truth saliency maps for training. We have designed an original two-step network interpretation method that supplies the binary salient motion segmentation. Finally, we recover the valued motion saliency map using a parametric flow inpainting method. Experimental results on real videos and comparison with existing methods will assess the performance of our method.

5.1 Introduction and Case studies

Motion saliency (MS) aims to highlight local motions departing from their surrounding context, thus prone to reveal a significant event. MS has many applications in computer vision. It can be useful in the navigation of autonomous vehicles to anticipate moving obstacles, or for public safety to trigger alert in abnormal situations. It can facilitate the subsequent analysis of videos where motion may play the key discriminative role.

In this chapter, we are concerned with the computation of salient motion maps from the optical flow only. Motion saliency (MS) cannot be formalized as an absolute notion. A motion is not salient in itself, but only with respect to its context, i.e., its surrounding motions or sometimes a predefined normal motion, and to a given application. A motion salient in a given context might not be salient in other contexts. In addition, we do not want to leverage the object appearance in the image. Indeed, the moving entity may not necessarily be a specific object in a scene, and its appearance may not be distinctive from the background content, e.g., someone in a crowd, a cell among others, a cloud among others.

We aim to design a general and versatile method that does not require any supervised training with ground-truth salient motion maps. Its main ingredient is to leverage the interpretation a frame-based motion saliency classification network. As aforementioned, we will handle two different cases of salient motion. The first one is the salient motion produced by independently moving objects in a scene observed by a moving camera.

Since we will rely on the optical flow only, a specific problem arises with a mobile camera: motion parallax attached to static objects being at the forefront of the scene and image motion of independently moving objects in the viewed scene both generate distinguishing patterns in the computed optical flow. Indeed, image motion depends on both the relative 3D motion between scene objects and camera, and on the object depth. We will solve the problem of motion parallax with unknown scene geometry and unknown camera motion.

The second case that we will investigate is salient motion issued from a distinctive motion within a coherently moving set. The set can be typically human crowd, animal herd, bird flock, vehicle traffic, or cell set. An example is the detection of crowd anomaly, where motion saliency comes from a person moving differently than the surrounding ones [69], [77], [123].

Our main contributions are the following. We estimate MS from the interpretation of a frame-based saliency classification network that takes optical flow (OF) as input. Our paradigm can accommodate a given form of motion saliency by simply training the frame-based classification network on the corresponding task. Moreover, our MS estimation is unsupervised, since it does not require any ground-truth saliency maps for training. In addition, we have designed an original two-step network interpretation method, which supplies the binary salient motion segmentation. Finally, we compute the motion saliency map using a parametric flow inpainting method.

The rest of the chapter is organized as follows. In Section 5.2, we will present our original framework involving several stages. We report experimental results with comparison to recent methods in Section 5.3. Section 5.5 will contain discussion and concluding remarks for this chapter.

5.2 Method Description

We proceed in two main stages to estimate motion saliency. First, we segment the salient motions from the two-step interpretation of the classification network that predicts the presence or the absence of salient motion in every frame of a video. Then, we compute the salient motion maps using a parametric flow inpainting technique. In the Results section, we will designate our method by the acronym NIMS (Network Interpretation for Motion Saliency).

5.2.1 Frame-based motion-saliency classification

We describe our frame-based classification network inspired from [124]. Basically, it involves the same layers comprising three convolutional blocks and a fully connected layer. However, it takes as input the optical flow field computed between two successive frames, and not a residual flow as in [124]. Thus, it is not necessary to compute any dominant motion resulting from the camera motion, which may introduce bias in case of complex background.

The classification network being shallow, the training is fast. For the first studied case of salient motion, it is trained to predict the presence of independent motion in a frame. It is thus implicitly able to discard motion parallax. In practice, we fine-tuned the network of [124] using the same training set as in [124]. For the second studied case, our network is trained from scratch to detect the presence of distinctive motion in crowds. We used the crowd synthetic dataset introduced in Section 5.3.3.

Our method is versatile in the sense that we just need to re-train the classification network on the task under study to deal with the salient motion of interest. The classification itself is supervised but the annotation process is extremely light. It boils down to labeling at the frame level, and in practice, it can be almost automatically completed, since the training dataset is composed of videos that were either fully dynamically salient or fully non salient.

We further modify the classification network for the interpretation. Following advice on applying interpretation techniques to deep neural networks [125], we modify the ordering of the inner layers as explained in Fig.5.1, which results in a functionally identical network but able to provide more operable interpretation maps.

Figure 5.1 – Modification of the ordering of the network inner layers. Left: Order for training and test. Right: Order for interpretation. Weights of each layer remain unchanged. For interpretation, Convolution and Batch Norm layers are merged into a functionally equivalent convolutional layer.

A mathematical justification of this modification can be formulated as follows. Considering $X : \{x_1, x_2, \cdots x_n;\ x_i \in \mathbb{R}\}$ a local neighborhood of the input, the results of the computation using the network training order (Max Pool, Batch Norm, Relu) is :

$$\tilde{x} \triangleq relu(\gamma \frac{\max_{x_i \in X}(x_i) - E_r}{\sqrt{V_r}} + \beta), \tag{5.1}$$

where $relu(x) = max(x, 0)$. E_r and V_r are representing respectively the running estimate of the expectation and the variance accumulated during training phase and used as a fixed value during test phase, and γ, β are the scaling and shift parameters of the batch norm layer learned during training. Thus, if the condition $\gamma \geq 0$ holds, which we verified in all our experiments, we can write :

$$\tilde{x} = \max_{x_i \in X}(relu(\gamma \frac{x_i - E_r}{\sqrt{V_r}} + \beta)). \tag{5.2}$$

This second computation is corresponding to the "Interpretation order" (Batch Norm, Relu, Max Pool) described in the paper. In cases where the running estimate of the expectation and the variance are not computed, we can proceed a first forward step through the training network to retrieve the expectation and variance value for a batch, and then, use those values in the interpretation network. Note that this whole manipulation is not necessary if in the original training order the normalisation layer is already adjacent to the foregoing convolutional layer. In this case, we can directly proceed to the interpretation step. After this step, the justification of the fusion between the Convolution and Normalisation layers is given in [125].

5.2.2 Inference of interpretation maps for salient motion segmentation

Since the classification network predicts correctly the dynamically salient frames and the non salient ones (with an average precision of about 90% in our experiments), we can deduce that it has learned to distinguish salient moving entities. Then, we can exploit this knowledge to locate the salient motions in the frame. Specifically, interpretation techniques allow one to generate attribution maps from the trained neural network. Those maps outline which parts of the input really contribute to the prediction. However, we are not dealing with images as usual but with 2D optical flow fields. Our goal is to determine for segmentation purpose which vectors of the optical flow field induce the prediction as dynamically salient frame.

Several techniques exist for network interpretation [126]. However, to the best of our knowledge, such a network interpretation has not yet been investigated for an optical flow input. An important issue is to extract interpretation information linked to the inner workings of the trained network, rather than to extract instance-specific structures information [126]. Several works such as [127] showed that in some cases, interpretation maps produced with a trained model could be visually similar to the ones generated by a random model. In our case, we did not observe such a behaviour, as shown by experiments reported at the end of Section 5.3 of this chapter. Integrated Gradients method [128] is a technique where maps are generated by cumulating gradients along a linear interpolation between a chosen baseline and a given input. Even though this technique delivers promising results on multiple datasets, results vary widely depending on the chosen baseline [129]. In the case of images, the default baseline is a black image that represents the absence of features. However, regarding optical flow, there is no obvious choice to represent the absence of salient motion. In our experiments, applying the Integrated Gradients method on optical flow fields mainly led to highlighting areas with large flow magnitude.

First step: point-wise computation of attribution maps

Instead, we have adopted the layer-wise relevance propagation (LRP) method [130]. It supplies less noisy attribution maps and removes the need to choose a baseline. This technique propagates a relevance score from the output layer to the input of the convolutional network. It relies on purposely designed propagation rules [130].

We have followed the structure presented in [126], applying the z^+ rule to all convolutional and fully-connected layers except for the first convolutional layer were we apply the z^β rule. The z^+ rule only takes into account positive weights during the propagation of relevance. The z^β rule propagates positive relevance through layers that take as input negative values, by using an additive term for the extremal admissible values of the input space. The expression of these rules are given below.

LRP rules. Following description in [126], [130], we compute the importance score from the prediction output to the input layer $L = 0$, and use the attribution scores (R^0) at this layer as the LRP attribution map. Z_+ rule, given in [126], [130], is applied for all linear and convolutional layers except the first :

$$z^+\text{-rule: } R_i^L = \sum_j \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} w_{i'j}^+} R_j^{L+1}. \tag{5.3}$$

Z_β rule, given in [126], [130], is applied for the first convolutional layer to deal with negative values in input flow :

$$z^\beta \text{ -rule: } R_i^L = \sum_j \frac{x_i w_{ij} - l w_{ij}^+ - h w_{ij}^-}{\sum_{i'} x_{i'} w_{i'j} - l w_{i'j}^+ - h w_{i'j}^-} R_j^{L+1}, \tag{5.4}$$

where x_i is the input value at layer L and $R^L \in \mathbb{R}_+^I$ the relevance score associated to this input map. Weights in the layer L are denoted w_{ij}. We define $w_{ij}^+ = w_{ij} * Id\{w_{ij} > 0\}$ and $w_{ij}^- = w_{ij} * Id\{w_{ij} < 0\}$.

In contrast to image intensities, the components of the flow vectors are not restricted to a predefined bounded range. Thus, we have to adopt a different normalisation technique. For each input flow x, we compute l and h as the minimal and maximal values of the channel corresponding to each flow component.

Max-pool layers are handled with a winner-takes-all strategy. In our case, we do not need to deal with normalisation layers as we merged them with convolutional layers.

We come out with attribution maps computed on a point-wise basis. However, these maps may be noisy, fragmented or even incomplete. To get an output of the network interpretation usable as salient motion segmentation maps, we need to introduce a second step in the network interpretation process.

Second step: transforming attribution maps into salient motion segments

Depending on the quality of the attribution maps and possibly the size of the salient moving objects, this second step may imply a light or a strong use of an additional representative flow information, but it is still driven by the attribution values. As described below, this additional flow information is straightforwardly derived from the input optical flow, and it corresponds to respectively a weighted mean flow vector or a hierarchy of affine motion models.

We proceed as follows for the light version of this second interpretation step. Using the square of the attribution values as weights, we compute over the whole frame the weighted average $\tilde{w}$ of the flow vectors, which is likely to adequately represent the salient motion in the frame. In case of several types of salient motion, several modes should be sought for. Then, for every point $p = (x, y)$ and its flow vector $w_p = (u_p, v_p)$, p belongs to a candidate region if $\alpha_1 < (w_p.\tilde{w}/||\tilde{w}||_2^2) - 1 < \alpha_2$. In practice, we take $\alpha_1 = -0.01$ and $\alpha_2 = 4$. We set asymmetric threshold values because flow orientation and magnitude are both involved making the problem a bit tangled.

The other more elaborate version of the second interpretation step relies on a hierarchical parametric motion segmentation. We take the optical flow as input, and we segment it by iteratively estimating affine motion models fitting the optical flow field, using a robust regression involving the Huber function $\mathcal{H}$. More precisely, we minimize the function:

$$\sum_{p \in \Omega_k} \mathcal{H}(a_1^k + a_2^k x + a_3^k y - u_p) + \mathcal{H}(a_4^k + a_5^k x + a_6^k y - v_p), \tag{5.5}$$

where Ω_k is the image part intervening at iteration k, and the a_i^k's the six parameters of the affine motion model computed at iteration k. At each iteration k, inliers are selected to form a layer. Inliers are pixels satisfying the constraint $\frac{1}{2}(\gamma_u^k|u_p - u_p^k| + \gamma_v^k|v_p - v_p^k|) < \beta$, where (u_p^k, v_p^k) are the coordinates of w_p^k, the flow vector given at p by the affine motion model estimated at iteration k. γ_u^k and γ_v^k are the standard deviations over the differences of the u and v components at iteration k. Those factors were introduced to avoid being affected by diverse motion magnitudes. In addition, it allows us to set the threshold β once and for all, and we take $\beta = 1.6$. Outliers are kept for the next iterations. We continue this procedure until only a small number of outliers remain (in practice 50 points), and we group them together to form the last layer. Following this iterative decomposition, each vector in the initial optical flow field belongs to one unique layer. Finally, we split each layer into connected components to form the candidate regions.

We still will have to select the salient moving regions among the candidate regions, whether it is for the light version or the elaborated one. This will be driven again by the attribution values as follows. As suggested in [126], the network interpretation must be primarily driven by the strength of attribution values rather than their spatial pattern. Therefore, we precisely take into account the attribution values to assign an attribution score to each candidate regions (i.e., segments of the optical flow). It is given by the ratio between the sum of the attribution values in the segment, and the size of the segment. Afterwards, we compute an adaptive threshold on the attribution score to filter out segments with the lowest scores considered as not salient. In practice, we use the score of the biggest region, assumed to correspond to the background, as threshold. This procedure allows us to select several salient segments if needed. Finally, we get a set $\mathcal{R}_{sal}$ of salient moving regions R_j.

5.2.3 Estimation of the motion saliency maps

We want now to estimate the motion saliency map. To this end, we will adopt a parametric flow inpainting. More specifically, we will extend within each segmented region of the set $\mathcal{R}_{sal}$ the optical flow surrounding this region, and compare it to the initially computed optical flow inside the region. To achieve it, it is easier to follow a parametric approach. First, we estimate an affine motion model fitting the external optical flow, i.e., the flow outside $\mathcal{R}_{sal}$. We use again the robust estimation of eq.(5.5), but here, it is applied to $\Omega \setminus \mathcal{R}_{sal}$ where Ω is the whole image. Then, we leverage this estimated affine motion model to inpaint the flow within all the regions R_j of $\mathcal{R}_{sal}$. The inpainted flow within each R_j is given by: $\forall p = (x, y) \in R_j, (u_p^{imp}, v_p^{imp}) = (\hat{a}_1^{sal} + \hat{a}_2^{sal} x + \hat{a}_3^{sal} y, \hat{a}_4^{sal} + \hat{a}_5^{sal} x + \hat{a}_6^{sal} y)$, where $\hat{a}_i^{sal}$ designates each of the parameters of the affine motion model estimated outside $\mathcal{R}_{sal}$.

The parametric inpainted flow allows us to infer the motion saliency map ϕ inside R_j from the flow gap $w_p^{imp} - w_p$, where w_p is the flow vector initially computed at p:

$$\text{for each } R_j,\ \forall p \in R_j, \qquad \phi(p) = 1 - \exp(-\lambda ||w_p^{imp} - w_p||_2), \tag{5.6}$$

where $\phi(p)$ values lie within $[0, 1]$ and $\lambda > 0$ modulates the visualization of the motion saliency map.$\phi(p)$ is set to 0 for all the pixels that do not belong to the set $\mathcal{R}_{sal} = \{R_j\}$. We use $\lambda = 0.15$ for all presented visualisations.

5.3 Experimental Results

5.3.1 Implementation details

Optical flow fields are computed using the RAFT method [56]. For LRP, we adopted a modified version of the implementation described in [126]. We used Captum to test Integrated Gradients and Scikit Learn to implement the Huber robust regression. For the first case study, we transferred the model and weights from [124] onto Pytorch framework, and fine-tuned the last fully connected layer using the original training dataset. We trained for 2 epochs, requiring approximately 20 min on a GPU GeForce MX150. For the second case study, we trained the network form scratch, which took 4.5 hours on a GPU GeForce RTX2080Ti. The average runtime per frame is 0.70s on a CPU 1.90GHz.

5.3.2 First case study: Independent scene motion

First, we want to objectively evaluate the accuracy of our method regarding the salient motion case of independent scene motion. Due to the lack of dedicated benchmarks, we make do with the VOS DAVIS2016 dataset [1] as is also done by VS methods. The DAVIS2016 videos involve one single independently moving object in the foreground.

Let us stress that the DAVIS2016 training set has not been used for training our classification network. In this experiment, we use the more elaborated second step of the interpretation process described in subsection 5.2.2. The bitmap of the salient moving regions R_j extracted by our method is compared to the ground truth. Results obtained with our method and other unsupervised methods are collected in Table 1. Our method outperforms comparable unsupervised methods on the VOS benchmark as BGM [54], TIS_0 [68], and FTS [104]. On our side, we do not resort to any postprocessing step and do not rely on appearance contrary to the CIS and TIS_s methods. Indeed, the CIS method involves an important post-processing stage and its performance without the postprocessing, evaluated with the publicly available code, drops to $\mathcal{J}\text{Mean} = 59.7$ on the complete DAVIS2016 dataset and to 59.2 on the validation set, that is, both lower than our scores. Besides, our method is penalized by the ground truth of DAVIS2016 focused on the primary object. For instance, the ripples on the water in the flamingo sequence and the foam under the kytesurfer (Fig.5.4) are recognized as salient motions by our method, which is correct.

1. https://davischallenge.org/index.html

	CIS [47]	TIS_s[68]	TIS_0[68]	BGM[54]	FTS [104]	NIMS (Ours)
$\mathcal{J}$ Mean ↑	- (71.5)	67.6 (62.6)	58.6 (56.2)	62.5 (-)	57.5 (55.8)	63.0 (60.2)
$\mathcal{F}$ Mean ↑	- (70.5)	63.9 (59.6)	47.5 (45.6)	59.3 (-)	53.6 (51.1)	68.2 (65.8)
$\mathcal{J}$ Recall ↑	- (86.5)	84.7 (-)	75.9 (-)	70.0 (-)	65.2 (64.9)	76.7 (73.2)
$\mathcal{F}$ Recall ↑	- (83.5)	78.5 (-)	48.8 (-)	66.2 (-)	57.9 (51.6)	80.1 (75.6)

Table 5.1 – Results on the complete DAVIS2016 dataset for several unsupervised methods (scores taken from [68] and [47]). In brackets on the DAVIS2016 validation set only. $\mathcal{J}$ is the Jaccard index (region similarity) and $\mathcal{F}$ accounts for contour accuracy. For further explanation on the evaluation metrics, we refer the reader to the DAVIS2016 website.

Figure 5.2 – Starting from the optical flow (left), we replace the flow in the selected regions by inpainting. When the tree-trunk region in the foreground is ablated, the frame is still classified as dynamically salient (middle). In contrast, ablating regions of the two moving people switches the frame as non salient (right). See the prediction score above each frame.

We want to further demonstrate the ability of our method to distinguish between independent motion and motion parallax. The latter being rarely present in DAVIS2016, we built a dataset of videos taken with a hand-held moving camera.Those videos are taken in an urban environment with complex backgrounds, where at the same time static objects in foreground induce motion parallax and other objects undergo independent motions. Examples of those videos are given in Figs.5.2-5.3-5.4 and in the supplementary material.

First, we assess the performance of the classification network in itself by performing an ablation test. We draw a mask around an area depicting distinguishable motion pattern and inpaint the flow within the mask as done in Section 5.2.3, hiding this prominent flow from the network. The procedure is illustrated in Fig.5.2. We can observe that removing only moving objects lowers the saliency score, and makes the network predict the frame as "non salient". Conversely, when we remove the static object causing motion parallax, the prediction of the network remains "salient frame" due to the presence of moving objects. It shows that the frame-based classification network truly bases its prediction on the presence of independent motion. Secondly, we visually evaluate the localisation of the salient moving regions. Fig.5.3 contains a comparison between the motion saliency

Figure 5.3 – Extracted salient masks, superimposed on original images, on three examples from our dataset and one from DAVIS2016 (*parkour* video). Top row : results from [47] using the available code and model (without the postprocessing step). Bottom row : corresponding results using our method (NIMS). CIS [47] detects the foreground static object due to parallax motion, while our method succesfully discards it.

Figure 5.4 – Results obtained with our method (NIMS). From left to right, two outdoor videos we acquired, five examples from the DAVIS2016 dataset, respectively, *parkour, libby, dance-twirl, flamingo and kite-surf* videos, and the *drive* video of the Complex dataset. Top row: one image of the video. Middle row: Optical flow in HSV color code; Bottom row: Motion saliency maps. The closer to yellow, the higher the motion saliency degree.

segmentation performed by our method, NIMS, and by the CIS method presented in [47]. We can note that our method is able to correctly extract the person moving as a salient moving region and not the pole, the tree trunk or the fence in the foreground. In contrast, the method [47] also segments them as independently moving objects.

Finally, we display samples of motion saliency maps in Fig.5.4. They were extracted from several datasets, respectively, our video dataset, the DAVIS2016 dataset and the Complex dataset [38]. The closer to yellow, the higher the motion saliency degree given by $\phi(p)$ in eq.(5.6). Let us stress that the static foreground objects are not found salient, whereas their optical flow is prominent since it conveys parallax motion. The computed salient motion maps are representative of the underlying motions.

5.3.3 Second case study: Distinctive motion in a crowd

In order to train our network and to evaluate the ability of our method to detect distinctive motion within a crowd, we built a dataset made of computer-generated sequences from 3D scenes depicting crowds in motion, using the public software ChAOS [2]. The resulting dataset is not trivial, since the salient moving persons are small and not always easily discernible. It is composed of 100 videos of about 250 frames each, taken by a camera in motion. In each video, the crowd of people is walking in a common, randomly chosen direction. In order to simulate distinctive motion, one to three people walking in a different direction are randomly added to some videos. Frames including a distinctive motion are dynamically salient.

The first step is to train the frame-based saliency classification network on this dataset for the new task. This randomly initialized network follows the structure defined in subsection 5.2.1, except that we used InstanceNorm layers instead of BatchNorm to tackle changes in input flow magnitudes. After training, the network is able to recognize frames containing a distinctive motion with 90.6% of accuracy on the test set. We apply the LRP interpretation to the classification network and extract attribution maps. The latter exhibit a low level of noise and mostly highlight salient motions. Therefore, we use the light version of the second step of the network interpretation process described in subsection 5.2.2.

Figure 5.5 displays samples of attribution and saliency maps computed with our method on videos of the crowd dataset. We can observe that these maps correctly delineate the salient motion in the crowd despite the small size of the salient area and the global camera motion. We have also processed several real videos of a similar kind. Visual results are again collected in Fig.5.5. We used the classification network trained on the synthetic dataset without any additional fine-tuning. We have processed the Wrong Way video that depicts a man walking against a crowd. As shown in the video frames displayed in the supplementary material, the salient motion is correctly segmented in the vast majority of them. The method fails in case of poorly visible salient motion. On the other hand, when a woman moves aside to avoid the opposite individual, she produces the main salient motion. We also processed videos from the UCSD Anomaly Detection dataset [3] fitting this second task. Again, correct salient motion masks are recovered, whether it be the vehicle or the pedestrian walking in the opposite direction. Let us note that in the

2. Crowd Animation Open Software : https://project.inria.fr/crowdscience/download/
3. http://www.svcl.ucsd.edu/projects/anomaly/dataset.html

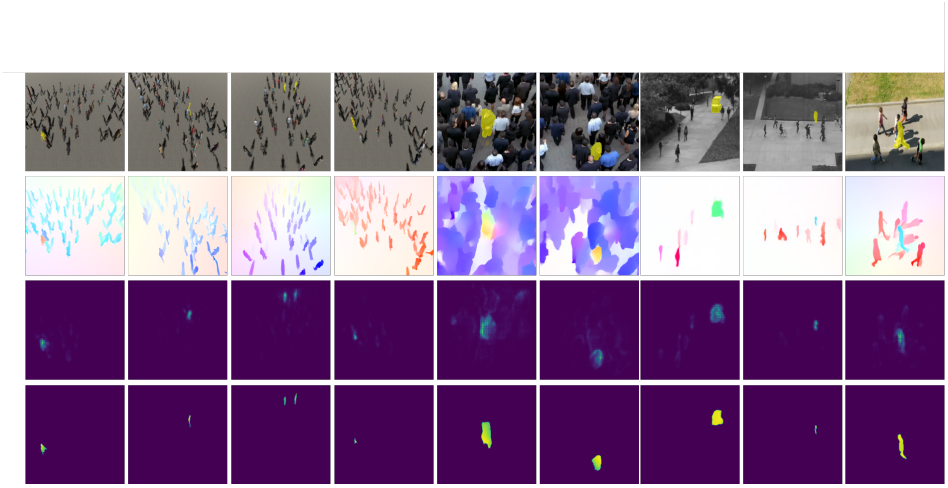


Figure 5.5 – Results obtained with our method (NIMS). Top to bottom row : one image of the video with the salient-motion ground-truth superimposed in yellow, optical flow displayed with the HSV color code, LRP attribution map, computed salient motion map. Left to right: four examples from the synthetic dataset (4th column: the outlier pedestrian is strongly occluded and only the foot is consequently recovered), two samples from the Wrong Way video at two distant time instants, two examples from the UCSD Anomaly Detection dataset, and an example from an outdoor scene acquired for the experiment.

later video, the anomaly regarding our task is precisely this pedestrian and not the cyclist as defined in the UCSD dataset benchmark.

In addition, we compared our method with a segmentation U-net network trained in a supervised way using the same synthetic dataset and ground-truth masks corresponding to the salient moving objects. The objective is to build a gold standard, and to achieve a quantitative evaluation on the synthetic dataset. We use again as evaluation metric the Jaccard index $\mathcal{J}$Mean. Let us note that high evaluation scores cannot be expected, because salient moving objects are frequently occluded. For a fair but strict evaluation, we compute it only on salient frames correctly classified as dynamically salient (since for non-salient frames the Jaccard index is 1, the ground-truth being empty). We get $\mathcal{J}$Mean $= 67.1$ for the supervised network, and $\mathcal{J}$Mean $= 53.6$ for our unsupervised method, which demonstrates that our method performs well, given the relatively small gap between the two scores.

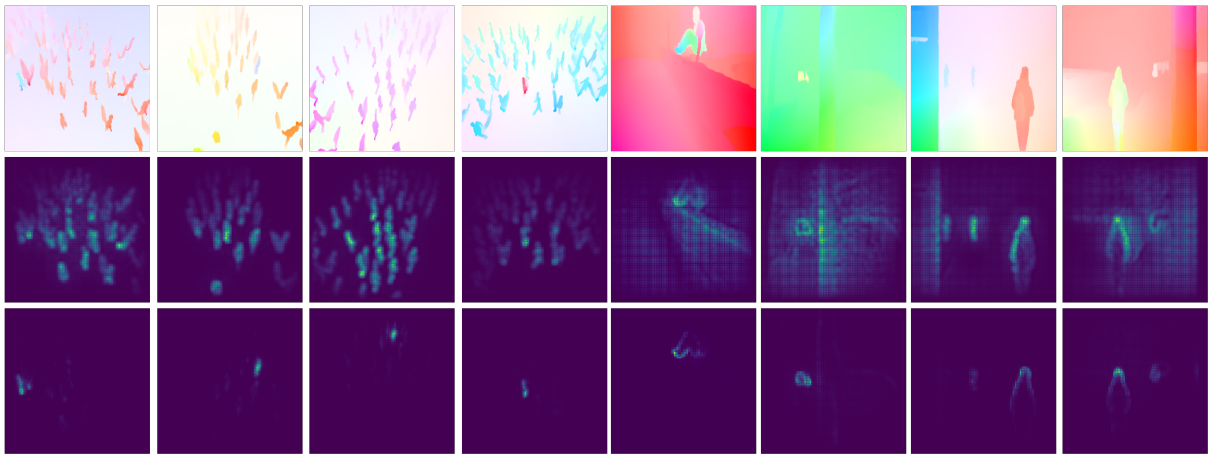


Figure 5.6 – From top to bottom : Input optical flow displayed with the classical HSV color code, LRP attribution map from the random classification network and LRP attribution maps obtained using our trained classification network.

5.3.4 Model randomization test

Interpretation methods can be sensitive to the structure of the network being analyzed and elements in the input images such as edges [127]. We want to ensure that our attribution (or interpretation) maps depend on the learned parameters of the network and not of intrinsic characteristics of our input flow maps only. Thus, we followed the recommendation in [127] and performed a model parameter randomization test. This test consists in comparing attribution maps obtained using our trained network and a random network. By visual inspection of Fig.5.6, we can observe that the attribution maps exhibit important differences. While attribution maps of the trained network are focusing only on points that exhibit salient motion, attribution maps of the random network highlight all points with an apparent motion indistinctively of its real saliency. This test confirms the pivotal role of training the network on a classification task to obtain meaningful attribution maps.

5.4 Combining LRP Segment Selection with Deep Learning Motion Segmentation

In this section, we combine our LRP-based method for motion saliency presented in this chapter and the temporally-consistent motion segmentation method described in

Segmentation	Huber	ST-Segmentation	
Selection	LRP	Except Biggest	LRP
blackswan	22.3	33.0	56.6
bmx-trees	52.5	58.5	56.3
breakdance	60.6	73.8	72.1
camel	63.2	87.2	85.6
car-roundabout	59.9	88.5	89.7
car-shadow	84.4	88.3	88.7
cows	62.5	87.4	85.5
dance-twirl	69.9	82.2	76.6
dog	64.9	81.3	74.8
drift-chicane	66.5	66.6	66.4
drift-straight	66.3	86.1	82.5
goat	27.1	25.1	27.7
horsejump-high	77.3	82.1	81.7
kite-surf	45.3	34.6	44.8
libby	72.7	40.5	66.3
motocross-jump	53.4	61.6	57.6
paragliding-launch	58.8	62.5	60.9
parkour	72.4	49.5	73.3
scooter-black	59.7	74.8	79.4
soapbox	64.1	88.9	85.4
Average	60.2	67.6	**70.6**

Table 5.2 – Comparaison of quantitative results for all the sequences of the DAVIS2016 validation dataset, obtained with three different combinations of motion segmentation and segment selection. Each row represents the average Jaccard score for the sequence, and the bottom row is the overall average Jaccard score for all the sequences. The first column includes the original results reported in Section 5.3, where we used Huber segmentation along with the LRP-based salient segment selection. The second column collects the results obtained using the deep-learning motion-segmentation method described in Chapter 3 and called ST-MS, with a simple heuristic where we select for evaluation all segments except the largest one as foreground. The last column reports the results obtained by combining the ST-MS method of Chapter 3 with this time the LRP-based salient segment selection. Red rows indicate sequences with a strong parallax motion, grey rows sequences with background parasite motion (e.g., water ripples), and the green rows sequences with complex motions to segment (e.g., articulated motions).

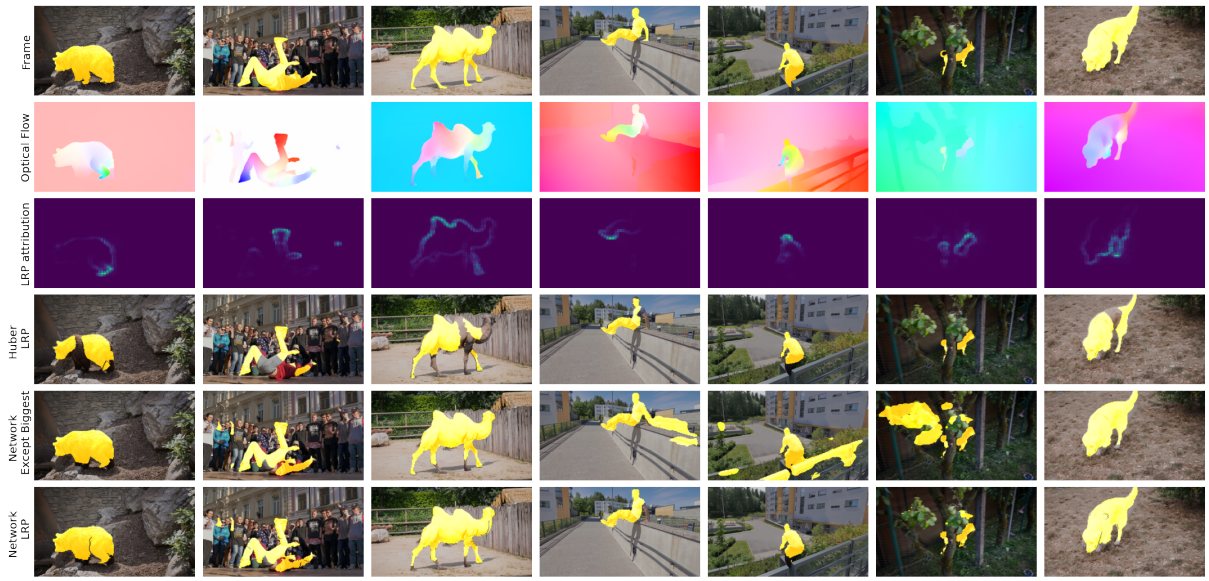


Figure 5.7 – Qualitative results on several sequences from the DAVIS2016 validation dataset. From top to bottom : RGB frame with GT segmentation mask overlayed in yellow, optical flow displayed with the HSV color code introduced in section 2.3, LRP attribution map, results obtained with Huber segmentation, results obtained with our ST-MS deep-learning method using the "except biggest" segment selection heuristic, and results obtained by combining the ST-MS method with LRP-based segment selection. All sequences are taken from the DAVIS2016 dataset, from left to right: *bear, breakdance, camel, parkour* (2), *libby* and *dog*.

Chapter 3 and called ST-MS. As a matter of fact, we worked on the LRP-based method before studying motion segmentation based on deep learning. Therefore, in the initial design of our LRP-based motion saliency method, we used an iterative flow segmentation algorithm based on the estimation of parametric motion models through a robust regression. This motion segmentation method worked well on simple optical flow fields, but struggled to adapt to more complex scenarios. Here, we replace this motion segmentation algorithm with the ST-MS network presented in Chapter 3, while keeping the rest of the motion-saliency approach identical to measure the gain we can obtain with the ST-MS method.

Quantitative results on the DAVIS2016 validation dataset reported in Fig.5.2, show the clear gain (70.6 vs. 60.2) obtained by using our elaborated ST-MS method over the one previously used in section 5.3. Conversely, comparing the second and third columns of Fig.5.2, we can also measure the performance improvement brought by applying the LRP-based network interpretation to select the right segments obtained with the ST-MS method. This is particularly important in scenarios where it is necessary to distinguish segments corresponding to motion parallax from those exhibiting independent motion (e.g., in *parkour* and *libby* sequences). In these cases, the "except biggest" segment selection algorithm indiscriminately selects all moving segments except the largest, which is considered background, whereas the LRP-based motion saliency method allows us to select only trully independent motion.

We also provide a qualitative comparison of the results obtained by these three approaches in Fig.5.7. We can observe that the Huber segmentation struggles especially on challenging optical flow fields representing articulated motion such as in the *bear, breakdance*, and *camel* sequences, while it performs well when there is an important difference between the object motion and the background motion (*libby, parkour* sequences). We can also see that the LRP-based segment selection process allows us to discard the motion parallax effect: in the *parkour* sequence, clear distinction between the apparent motion of the barrier from that of the running man, and in the *libby* sequence, between the apparent motion of the tree from that of the dog.

5.5 Partial Conclusion

We have defined an original, versatile and efficient method for the estimation of salient motion maps based on the interpretation of a simple frame-based classification network.

It only requires the optical flow as input. The computation of the salient motion maps is unsupervised as it does not require any ground-truth on the segmentation maps. To tackle a given motion-saliency task related to a given application, it is sufficient to re-train the classification network, the overall framework remaining unchanged. We have demonstrated the performance of our method on two different case studies. Our method was thus able to disentangle parallax motion and independent scene motion without any 3D information. It can also handle distinctive motion in a crowd. Experimental results demonstrated that we are able to compute accurate and reliable salient motion maps which convey rich, readily available information on the nature of the salient motion, through the flow gap of the inpainting step, and not only the degree of saliency. Finally, combining the LRP-based motion saliency method and the ST-MS method of Chapter 3 provides the best results.

Nevertheless, this motion saliency approach requires an external motion segmentation stage to delimit regions exhibiting salient motion. In the next chapter, we will propose an integrated approach for salient motion segmentation based on adversarial learning.

Chapter 6

Adversarial Learning to Locate Salient Motions

6.1 Adversarial Approach for Saliency Segmentation

In this chapter, we aim to build a method for learning to segment regions of salient motion in optical flow fields, based on the predictions of a classification network. This work is an initial study to design an alternative to the method presented in Chapter 5, where we want to fuse the segmentation and interpretation steps to directly output the salient motion segmentation from the optic flow field. Instead of relying on gradient-based interpretation techniques such as LRP, we train a segmentation network to locate areas that, when modified (or perturbed), change the prediction of the classification network. This approach has several advantages. First, compared to the gradient-based interpretation approach, it is not sensitive to biases induced by the classification network or the input data. Secondly, after training, the segmentation network produces the motion saliency mask directly from the optic flow field, and is thus independent of the classification network at the inference step. Finally, unlike the gradient-based method presented in Chapter 5, it does not depend on post-processing techniques to extract the segments of salient motion.

In the first part of this chapter, we provide a brief overview of related work using perturbation-based approach for neural network interpretation. In a second part, we provide a description of our framework and our loss function based on adversarial learning. Finally, we present preliminary experimental results, mainly on the second motion saliency task presented in Chapter 5, i.e., discriminating singular motion within a coherently moving set.

6.2 Related work on perturbation-based network interpretation

In Chapter 5, we presented network interpretation methods based on gradients to determine the image areas significantly contributing to the prediction of the network classification. In this chapter, we focus on methods that perturbate a region of the input of a neural network to change the prediction of the classification network. In [131], the authors train a network to segment salient areas in an image based on a differentiable classifier. Their loss function compares the prediction of the classifier when only the salient area is perturbed (using either a blurred version of the image or a random colour image with gaussian noise) to the prediction when everything except the salient area is pertubed. They also use total variation (TV) regularization to produce smooth segmentation masks and introduce a loss term on the mask size. Using a similar approach, the authors in [132] propose a method for training a segmentation network that localizes salient regions. Instead of using a loss term on the size of the masked region, they design a loss term based on information bottleneck to control the information complexity of the masked region. In practice, they compute it from the reconstruction loss of a variational autoencoder [133], taking as input the binary mask and the masked image.

The problem with perturbing regions of the input image using classical techniques, such as blurring or adding Gaussian noise, is that it can produce distribution images that are poorly handled by the classification network. To perturb input images in a more realistic way, [134] introduce a method where they train a generator network that can generate a realistic patch from its neighborhood. Using this method, they can compare the classifier output for an image after patch removal and obtain a feature importance map. Their method is highly dependent on the quality of the patch generator network, which can be difficult to train. In addition, they evaluate the importance of each patch sequentially, which may fail to detect interdependencies between patches. In some cases, removing a single patch may not affect the network prediction, but removing a set of patches may have a strong impact.

To our knowledge, there is no method that trains a motion saliency segmentation network using optical flow as input from a classification network. The advantage of our approach is that it is easier to obtain a realistic perturbation of the optical flow field, since it contains less details than RGB images. We mean that optical flow is globally smoother. In our method, we simply resort to parametric optical flow inpainting for this step.

6.3 Adversarial Network for Motion Saliency Segmentation

6.3.1 Overall framework

The core idea is to train a network to segment areas that are determinant for the prediction of a frame-based motion saliency classification network. The overall framework is illustrated in Fig.6.1. In order to describe this framework in details, we first introduce its core components :

- $f \in \mathbb{R}^{W \times H \times 2}$ is the input optical flow field. A salient flow field contains at least one salient motion and a non-salient flow field contains no salient motion.
- $\phi : \mathbb{R}^{W \times H \times 2} \rightarrow [0, 1]$ is a pretrained classification network, it takes as input an optical flow field and outputs its probability of being salient: $p_\phi(\text{salient}|f)$.
- $g_\gamma : \mathbb{R}^{W \times H \times 2} \rightarrow [0, 1]^{W \times H}$ is a segmentation network parametrized by γ, it takes as input an optical flow field and returns a probabilistic mask. We denote the predicted mask by $m \triangleq g_\gamma(f)$.
- $\omega : \mathbb{R}^{W \times H \times 2} \times [0, 1]^{W \times H} \rightarrow \mathbb{R}^{W \times H \times 2}$ is an inpainting operation. It takes a input a mask m and an optical flow field f and replaces the flow inside the mask using a parametric flow field computed on the flow field outside the mask. Its aims to produce a natural-looking flow while removing the masked area. We call this newly produced flow field $\omega(m, f)$ the "perturbed flow".

Our goal is to train the weights γ of a segmentation network g to produce a mask $g_\gamma(f)$ that after inpainting step results in a perturbed flow field $\omega(g_\gamma(f), f)$ classified as non-salient by ϕ.

6.3.2 Loss function

The objective described above can be specified by the loss function $\mathcal{L}$ for an optical flow field f as:

$$\mathcal{L}(f, \gamma) = p_\phi(\text{salient}|\omega(g_\gamma(f), f)), \tag{6.1}$$

with $p_\phi(\text{salient}|\omega(g_\gamma(f), f)) = \phi(\omega(g_\gamma(f), f))$. Indeed, minimizing this loss means that the probability of the pertubed flow being salient tends to 0, that is, we have correctly removed the area with the salient motion.

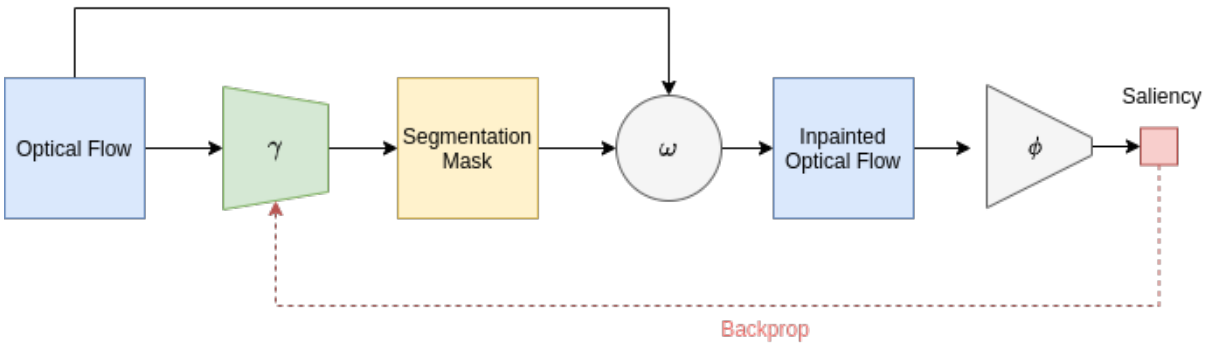


Figure 6.1 – Overall framework of our adversarial saliency segmentation method. g_γ represents the segmentation network that outputs the saliency mask, ω the inpainting operation that fills the flow inside the mask using flow information outside of it, and ϕ is the classification network that predicts if the inpainted flow is salient.

However, there are two important issues with the loss function (6.1). First, a trivial solution to minimize $\mathcal{L}$ is to predict $g_\gamma(f) = 1^{W \times H}$ (which means that all the points are predicted as salient) for all input optical flow fields f, which results in perturbed flow fields exhibiting no motion and thus classified as non salient. In order to avoid that, we add a term $\mathcal{S}(f, \gamma) = ||g_\gamma(f)||_1$ that penalizes masking large areas of the input. Secondly, the loss function (6.1) leads to directly minimize $\phi(\omega(g_\gamma(f), f))$, although we are only concerned with whether the perturbed flow is classified as non salient. Therefore, the constraint $\phi(\omega(\gamma(f), f)) < 0.5$ is sufficient, as the classification network has been previously evaluated on its global accuracy and not its ability to predict the correct saliency probability values for each input.

Taking into account those issues, we can reformulate our objective as to find the smallest region (possibly consisting of disconnected components) in the input flow that, when perturbed, changes the prediction of the classifier such that the flow becomes classified as non-salient. We can formulate this objective as follows:

$$\min_\gamma \mathcal{S}(f, \gamma) \text{ with } \phi(\omega(g_\gamma(f), f)) \leq 0.5. \tag{6.2}$$

Using the method of Lagrange multipliers [135], we can reformulate this constrained optimisation problem into an unconstrained problem:

$$\mathcal{L}(f, \gamma, \lambda) = \mathcal{S}(f, \gamma) + \lambda\, \sigma(\phi(\omega(g_\gamma(f), f)), 0.5), \tag{6.3}$$

where $\sigma(x, \kappa) = x$ when $x \geq \kappa$ and $\sigma(x, \kappa) = 0$ when $x \leq \kappa$. The parameter λ is the

Lagrange multiplier enforcing the constraint.

As we want to train the segmentation network g_γ, we need to find a gradient-based optimisation method to solve the above problem. Authors in [135] propose to solve this optimisation problem using the "Basic Differential Multiplier Method" (BDMM), where one performs a gradient descent on the parameters γ and a gradient ascent on λ, resulting in updates of the form:

$$\gamma_{t+1} = \gamma_t - \alpha \nabla_\gamma \mathcal{L}(f, \gamma, \lambda_t), \tag{6.4}$$

$$\lambda_{t+1} = \lambda_t + \alpha \nabla_\lambda \mathcal{L}(f, \gamma_t, \lambda), \tag{6.5}$$

where α is the learning rate. We train the network on a dataset $\mathcal{D}$ of optical flow fields. Thus, the overall loss function is:

$$\mathcal{L} = \sum_{f \in D} \mathcal{L}(f, \gamma, \lambda). \tag{6.6}$$

6.3.3 Flow inpainting

The goal of the flow inpainting step $\omega(m, f)$ is to fill the areas within the mask $m = g_\gamma(f)$ using the information of the optical flow field f. We use parametric motion models as the ones described in Section 2.4.1. Let us assume at this point that we have only two types of motions at most in the flow field: background motion and possibly salient motion. Nevertheless, salient motion may be found in several areas of the frame, meaning that the mask corresponding to salient motion may comprise disconnected components. Let us add that these salient areas do not necessarily undergo the same motion, but their motions are all salient with respect to the surrounding (background) motion.

First, we compute the parameters θ of the parametric motion model from the area outside the mask according to:

$$\theta_{inp} = \arg\min_\theta \sum_{p \in \Omega} (1 - m(p)) ||f(p) - \tilde{f}_\theta(p)||_1, \tag{6.7}$$

where Ω is the image grid, and $m(p)$, $f(p)$ and $\tilde{f}_\theta(p)$ respectively the saliency prediction, the optical flow vector, and the flow vector of the parametric motion model, at point p.

Then, we inpaint the flow within the masked area as :

$$\omega(m, f) = (1 - m) \odot f + m \odot \tilde{f}_{\theta_{inp}}, \tag{6.8}$$

where the operator $\odot$ denotes the Hadamard product between the two matrices.

6.3.4 Multi-mask extension

The issue with the inpainting technique presented in Section 6.3.3 is that it assumes the presence of at most two types of motions: background motion and optional salient motion. In the case where there is a greater number of motion types in the optical flow field, it would be impossible to compute correct motion model parameters θ_{inp}, since the background area would involve several motions. This situation occurs for instance in the Chaos dataset presented in Chapter 5. We have the background motion, which is null when the camera is static, the (coherent) crowd motion, and the salient motion of one or a couple of pedestrians walking differently than the crowd.

This motivated the introduction of a multi-mask extension of our technique where g_γ is not only predicting a salient mask, but also, a background mask and a set of subsidiary masks. In this case, the segmentation network no longer outputs one mask, but a set of K probabilistic masks such that $\forall\ p \in \Omega, \sum_{k=1}^{K} g_\gamma^k(f)(p) = 1$. The layers representing the background and the salient masks are hardcoded as being respectively the first and the second ones of the segmentation network output. In the rest of this chapter, we refer to the salient mask as $m_s \triangleq g_\gamma^0(f)$, the background mask as $m_b \triangleq g_\gamma^1(f)$. Let us stress that at this point we are talking about the functional background, not the physical background, that is, the area used to compute the inpainting parametric motion model. We can also refer to each mask using its index $m_k \triangleq g_\gamma^k(f)$.

Modified flow inpainting

In this setting, the inpainting step takes a different form:

$$\theta_{inp} = \arg\min_{\theta} \sum_{p \in \Omega} m_b(p) ||f(p) - \tilde{f}_\theta(p)||_1, \tag{6.9}$$

$$\omega(m, f) = (1 - m_s) \odot f + m_s \odot \tilde{f}_{\theta_{inp}}. \tag{6.10}$$

Coherence loss term

In this multi-mask extension, we can introduce the segment-wise flow reconstruction loss term presented in Section 2.12. This has the advantage of training g_γ to output motion coherent masks that we can use downstream for other applications. It is also a

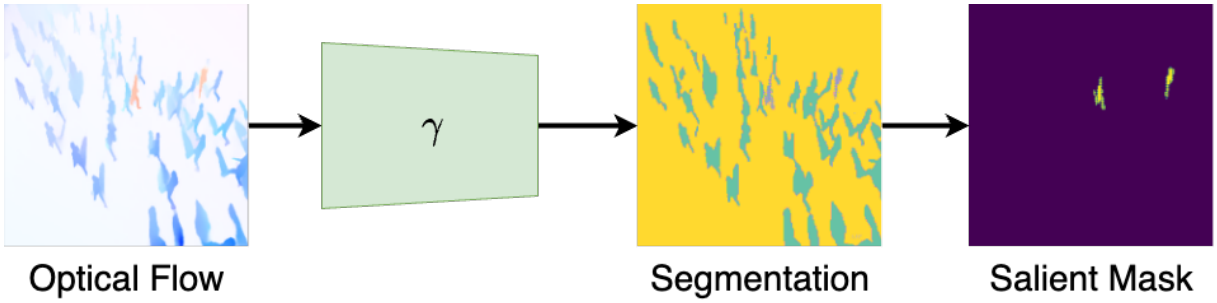


Figure 6.2 – Inference step of the adversarial network. From left to right: the input optical flow field in HSV color code, the segmentation network, the segmentation with all masks (background in yellow, salient in blue, other motions in green), the salient mask (extraction of the blue mask).

pretext task we can leverage to improve our network training. Let us recall its expression:

$$\mathcal{L}_r(m, f) = \frac{\sum_{p\in\Omega}\sum_{k=1}^{K} m_k(p)\|f(p) - \tilde{f}_{\theta_k}(p)\|_1}{\sum_{p\in\Omega}\|f(p)\|_1}, \tag{6.11}$$

$$\text{with } \theta_k = \arg\min_{\theta} \sum_{p\in\Omega} m_k(p)||f(p) - \tilde{f}_{\theta}(p)||_1. \tag{6.12}$$

For the choice of K, we can take any number greater than the maximum number of motions we expect to observe in the video sequence.

6.3.5 Inference step

In the inference step, we provide an optical flow field to the network, and it outputs the segmentation with the four masks (background, salient objects, other motions). In the next Section on experimental results, we will display only the salient mask, since it is the one of interest for our task. Contrary to the training time, it is not necessary to use the classification network to compute parametric motion models at test time, which makes the network efficient.

6.4 Experimental Results

6.4.1 Implementation details

For the segmentation network, we use the same U-net as described in Chapter 2. We output four motion segments, the first one is by default dedicated to the background mask (support used to compute the parameters of the background motion model) and the second one is devoted to the salient motion. The remaining two masks are dedicated to representing non-salient motion segments that are different from the background. To implement the gradient ascent in BDMM, we use the Gradient Reversal Layer introduced in [136], which allows us to perform both optimization steps described in 6.4, référence à inclure at the same time. The computation of parametric motion models during training is also the same as presented in Chapter 2.

We use the Adam optimizer with a learning rate of 10^{-6} and a linear decay schedule starting at epoch 100. Since the loss depends on λ, which is increasing, it is not necessarily decreasing. For the network model selection, for now, we just choose the model of the last training epoch and use it for the evaluation. We keep the choice of the criterion for a stopping criterion on the validation set for future work.

Regarding the compute time at inference time, we can expect the same results as in Chapter 2, since we use the same network. At training time, the classification network introduces a small overhead.

6.4.2 Detecting anomalous motion

At this stage, we report preliminary results.

Regarding the first motion saliency task, i.e., detecting independent motions in a scene observed by a mobile camera, results are not that convincing so far. Indeed, as illustrated in Fig.6.3, if the adverse network is able to detect the salient moving objects, it tends to extract only the boundaries of the independent moving objects. Further investigation will be needed.

Therefore, we will focus in this section on the second motion saliency task concerned with discriminating singular motion within a coherently moving set.

Figure 6.3 – Results regarding the task of independent motion detection on DAVIS2016 dataset. From left to right: *breakdance, car-roundabout, parkour, camel, dog* videos. From top to bottom: RGB frame, optical flow field (HSV color code), predicted salient motion mask.

Results on Chaos dataset

As we can see, the results on the Chaos sequences are quite good. The network detects most of the abnormal motions, and delivers a blank mask when there is no salient motion. Since the segmentation used for this task in Chapter 5 worked well, the results are on par with the previous ones, although in this case the network does both motion segmentation and salient moving object localization at the same time. We show examples of results in Fig.6.4.

Results on real sequences

To process real sequences, we use our network trained on Chaos dataset without any finetuning. Results on the *Wrong way* sequence are less convincing. It seems that the network is not able to segment objects well in these sequences, which is understandable since it was only trained on simulated optical flows. The results on videos from the UCSD datset, are better. In this case, the network manages to segment the salient motions well. To further improve these results, it would be interesting to train the network, or at least the U-net segmentation backbone, on real data. We show examples of results in Fig.6.5.

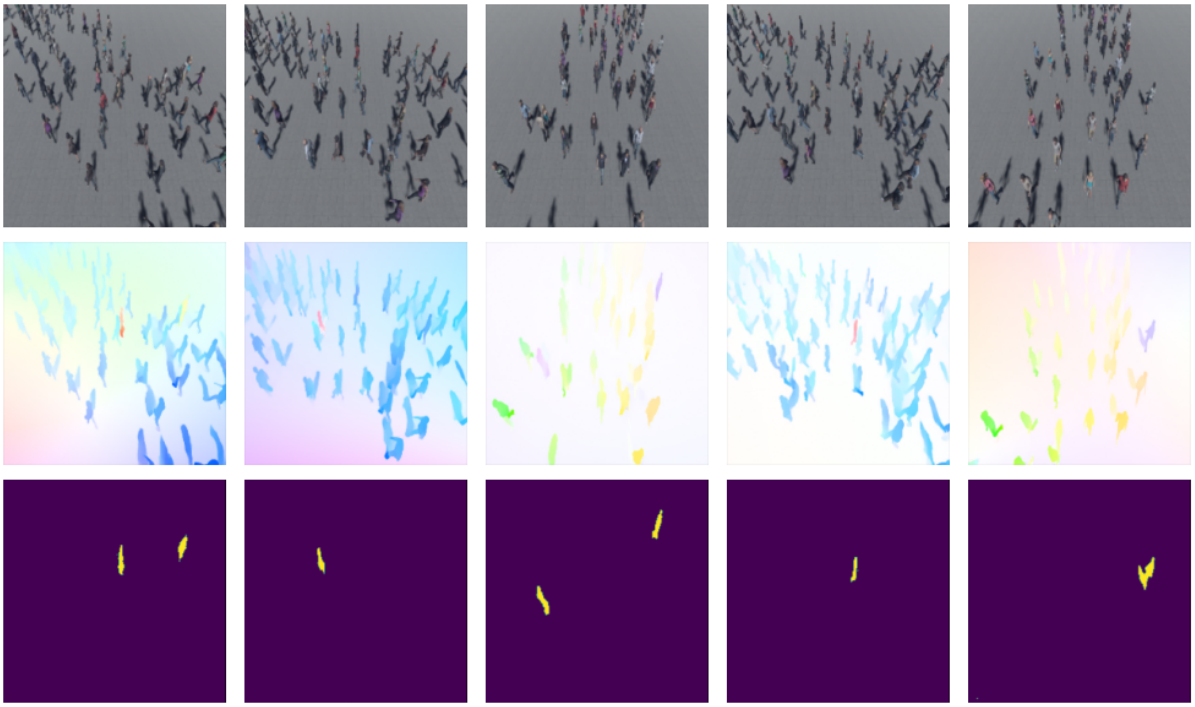


Figure 6.4 – Results on simulated sequences from Chaos dataset. Top to bottom : RGB frame, optical flow field (HSV color code), predicted salient motion mask.

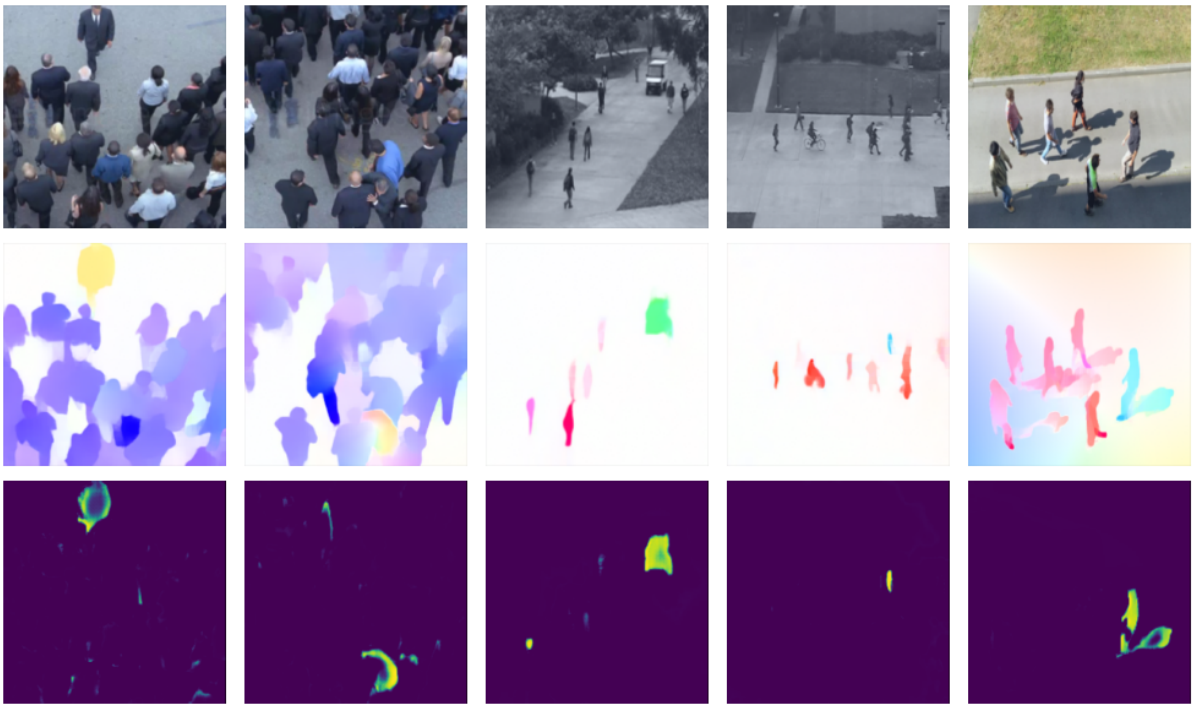


Figure 6.5 – Results from left to right on *Wrong Way* sequence (at two different time instants frames), on real sequences of UCSD dataset (two examples), and on an outdoor scene acquired for the experiment (one example). From top to bottom: RGB frame, optical flow field (HSV color code), predicted salient motion mask.

6.5 Partial Conclusion

In this chapter, we have presented an adversarial method for training a salient motion segmentation network from a pre-trained classification network. Our loss function is based on treating the problem as a constrained optimization and the Lagrange multipliers method to formulate the loss function. We have reported encouraging results on simulated data, validating the intuition behind our paradigm. Yet, the network does not perform well enough on real data.

Several modifications could help improve the results on real-world data. First, it would be interesting to have a variable number of masks to represent salient motion. In this first implementation, we have only one mask representing salient motion. Since this mask is affected by the coherence motion loss term, it restricts the prediction of salient motion to a single motion model. Experiments in Chapters 2, 3, and 4 demonstrated that the aggregation of multiple motion masks is key to represent salient motions on the VOS dataset. An interesting first perspective would be to allow multiple salient motion masks and aggregate them to extract the whole salient motion.

Second, it might be interesting to decouple the segmentation and the motion saliency classification network. Indeed, the motion segmentation network takes much longer to train and can be trained on a wider variety of optical flows, since it does not require a frame classification network. This can affect the results on hard to segment sequences such as *Wrong way*, where the flows to segment at test time are very different from the flows encountered in the training set. This improvement is hard to implement with convnets, but much easier to implement with the Maskformer architecture.

Third, we could introduce the temporal dimension in our segmentation, as it could improve the distinction between salient and non-salient motions. The optimal setting would be to introduce the temporal dimension in both the frame-based classifier and the segmentation network. Segmenting a volume of frames enforcing temporal consistency and then doing the classification for each frame independently could be a relevant first improvement.

General Conclusion

Contributions

In this thesis we investigated two axes for motion analysis. The first one is concerned with unsupervised motion segmentation, where we have designed and trained a network to decompose an input flow field into coherent segments. The second one deals with the localization of salient motions, where we have developed techniques to identify regions of the input flow field that are critical for a downstream classification task.

Unsupervised motion segmentation

In this part, we have developed a series of works to partition an input optical flow field into coherent motion segments.

In a first work, we introduced a framework to train a neural network to partition an optical flow field into coherent segments without relying on annotated data. Our loss function is based on the EM algorithm and the use of parametric motion models to represent the flow within each segment. We succeeded in training a network capable of segmenting unseen optical flow fields into a given number of coherent parts. Our method gives good results on several video object segmentation datasets (DAVIS2016, FBMS59, SegTrackV2, MoCA), while being fast at test time.

In a second work, we extended this framework to handle a volume of three optical flow fields segmented with coherent labels. To enforce temporal consistency, we coupled the loss from our first work on motion segmentation with a temporal consistency loss that enforces that the labels are consistent across the output segmentation volume. We showed that this work improves the robustness of the segmentation to noise in the input optical flow field, and that with a simple post-processing step we can produce a segmentation with consistent labels over a long sequence.

In a third work, we deepened our idea of segmenting long optical flow sequences by extending our previous method and defining a ent-to-end model to segment a full sequence. First, we introduce the use of splines to represent the temporal evolution of the parametric

motion model that represents the optical flow within each segment. Second, we moved from 3D convolutional networks to an architecture based on transformers to account for longer temporal context and introduce feature interactions across the sequence. This allowed us to have a network capable of segmenting an entire sequence of optical flow fields in a single pass, providing coherent labels across time. It achieves competitive results on video object segmentation benchmarks and is fast at test time.

Localization of salient motions

We have designed methods for localizing salient areas in optical flow fields. We define salient areas as those that affect the prediction of a saliency classification network, which identifies whether an input optical flow field has salient motion.

In a first work, we developed a fast and versatile technique based on the interpretation of a simple frame-based classification network. We use the LRP method to compute saliency, which is then combined with external motion segmentation to obtain saliency maps. Our results demonstrate the adaptability of this method to various saliency scenarios, only requiring a retraining of the classification module to adapt to a given task. We evaluated this method on two different motion saliency scenarios. The first scenario involved disentangling motion parallax from independent scene motions in videos recorded with a moving camera, while the second scenario focused on the localization of distinctive motions in a crowd.

In a subsequent study, we maintained the goal of predicting important areas for the prediction of a frame-based saliency detection network. However, we adopt a perturbation-based approach instead of the gradient-based method used in our prior work. This approach involved training an adversarial convolutional network to segment salient areas in the input optical flow field. A loss function was used to minimize the output of the classification network. This alternate method is appealing since it merges saliency detection and motion segmentation. However, further development is needed to improve results on real video sequences.

Possible improvements on unsupervised motion segmentation

Train on longer sequences

One possible improvement would be to enhance the training procedure for long-term motion segmentation by including longer sequences at training time. This could improve

the ability of the network to produce temporally consistent segments and help develop features that represent long-term motion. Although this is challenging as it would require modifying the training procedure to deal with the memory overload induced by training on long sequences, we could also modify our spatio-temporal attention operations to be less computationally expensive.

Variable number of segments

In the approaches we have designed so far, the number of segments produced by the network must be fixed at training time and cannot change during inference. An improvement of our method would be to allow the user to choose the number of segments to use at test time. We have done some experiments replacing our transformer attention with a slot attention [120] mechanism, although we encountered difficulties to get reproducible results and to ensure the convergence of the network. A somewhat related idea would be to train the network to do hierarchical segmentation, as done in [33], where related segments are grouped together. This would allow the users to choose which level of precision to use for their task.

Include appearance

Including appearance could allow to improve segmentation on parts of an object that are currently not moving (e.g. the moving objects stops at some point). However, as mentioned in the introduction, using appearance may also bias the segmentation towards visually salient objects if they predominate in the dataset. One option would be to simultaneously train a branch that takes object appearance as input and one that takes the optical flow field as input using our motion coherence criteria, and add a constraint that makes the two predictions match. At inference time, we could combine the predictions from the two modalities to ensure that salient static objects are not segmented. Another option would be to learn the appearance of the moving object in a video and use it to refine our motion segmentation.

Remove dependency on optical flow computation

So far, we provide the optical flow field as input to our segmentation network. This makes our network dependent on the quality of the flow field. We could improve our method to make it independent of the optical flow field method. A first improvement

would be to replace the optical flow field with a sequence of RGB frames as input to the network. This would require changing the structure of the network to implicitly compute correlation maps, as optical flow methods do, which would increase the computational load of our network. This would remove the dependence on the optical flow field at test time. However, since we need optical flow fields to compute the loss function, it would be dependent on the optical flow computation method at train time. A second improvement would be to modify our loss function with a reconstruction term that measures the warping error induced by the parametric flow field. Together with the input change mentioned above, this would make our segmentation method independent of optical flow field computation at both training and test time allowing us to learn motion segmentation without ever having to compute explicitly an optical flow field.

Possible Improvements on the localization of salient motions

Temporal Consistency

A salient object can be expected to remain salient for a few frames. To incorporate temporal consistency into salient object detection and segment a salient object over multiple frames, our method can be expanded to include video sequences. Extending our method to video sequences could also enable us to detect more complex forms of motion saliency that are not visible with a single instantaneous motion but are observable in an abnormal sequence of events.

Anomaly detection

In our previous work, we applied our method to obtain the interpretation of a saliency classification network that was trained in a supervised manner on a database of salient and non-salient optical flows. An interesting extension would be to apply our technique to an anomaly detection network trained in an unsupervised manner (e.g. one class SVM, teacher student approach as in [137]). This would eliminate the need for explicit supervision by presenting only non-salient images to the network.

Larger Research Perspectives

Self-supervised learning of appearance features from motion

Learning appearance representation from videos is an idea that has been around for a long time [3], [4], [138]–[142], and has shown interesting results as a pre-training task for supervised learning. This is an attractive alternative to other self-supervised appearance representation methods, such as [6], as it could learn the representation directly from videos without requiring an object-centric dataset. It would thus allow to deal with a greater variety of objects or image modalities. With the recent improvement of optical flow methods [56], [143] and the development of novel network architectures [120], [144], it seems that motion-based self-supervised representation learning could be a promising research direction in the future. An interesting perspective related to our work would be to learn appearance features at the sequence level, exploiting the temporal consistency of motion segmentation, as we did in Chapter 4. This could lead to the development of appearance features that are robust to temporal evolution (exposure changes, deformations, different camera angles). For example, a motivating idea would be to "capture" the object appearance during the first frames of the video and try to retrieve the object mask later in the video, verifying the quality of the prediction using a motion consistency metric.

Learning representations for complex motion

Action recognition is central to a number of tasks in computer vision and has an important impact because it is used in a wide range of applications. As mentioned in the introduction, existing action recognition methods exploit both motion and appearance cues. However, using images as input to action recognition models can raise a number of issues. First, technical issues, such as the difficulty of performing domain adaptation from simulated data in the image modality compared to performing it in the motion modality, the greater variability depending on context in images compared to motion representations. Second, social issues such as the occurrence of appearance-based bias in deep learning models based on images [145], [146]. Furthermore, recent camera developments such as event-based cameras [147] or on-sensor acceleration [148] make optical flow computation computationally cheaper. This context motivates the development of action recognition methods based solely on motion, and especially unsupervised methods that could learn to handle a large range of complex motion without expensive human annotation. However, the whole difficulty of unsupervised action recognition would be to learn

motion features that are complex enough to describe high-level motions and generalizable enough to adapt to natural variations in motions (i.e., not everyone claps their hands in exactly the same way, but our brain is able to recognize this action in a wide range of contexts). Another challenge is to group hierarchical motions together to associate them with the same cause (e.g., articulated motion).

In our work on long-term motion segmentation, we learn the representation of motion groups over a sequence as "refined queries" in the maskformer architecture. These queries allow us to build motion segmentation over a sequence using a cross product with the feature map. Thus, it seems that they contain global information about the motion of each group along the sequence. An interesting investigation would be to measure if these queries are informative about the high-level representation of motion, in which case it would make unsupervised long-term motion segmentation a great pre-training task for action recognition or clustering. Future work could focus on defining novel tasks or training procedures to improve the expressiveness of our motion representations, and finding ways to measure this expressiveness.

Dance movements and notations like the one we introduced in Fig.3 seem to be a fascinating benchmark for these tasks, as they describe a precise and complex movement that will be reproduced by a wide range of interpreters. The Laban notation is quite restrictive, but other notations [149], [150] offer more freedom to the dancers. A motion representation of videos that naturally clusters a set of dance moves would be, in my opinion, an impressive first step towards unsupervised action recognition.

Chapter 7

Appendix

7.1 Appendix A: EM-driven Motion Segmentation

7.1.1 Variational inference

In this subsection, we describe an alternative mathematical approach to lead to our loss function, based on the variational inference framework [151].

We consider a volume of optical flow fields $f \in \mathbb{R}^{2\times W\times H}$, the spatial grid $\Omega \in \mathbb{R}^{W\times H}$. We denote $f(i) \in \mathbb{R}^2$ the flow vector at site $i \in \Omega$. We assume that we can decompose the flow field into a set of K segments, each one exhibiting a coherent motion. Flow vectors within a given segment k are represented by a smooth parametric motion model parametrized by ϑ_k. Variable z_i conveys the motion segmentation: $z_i^k = 1$ if site i belongs to segment k, $z_i^k = 0$ otherwise. z and ϑ are latent variables, and f is the observed data. Following reasoning developed in [151], we introduce an approximate distribution over segmentation and motion model parameters:

$$q(z,\vartheta|f) = q(z|f)q(\vartheta) = (\prod_{i\in\Omega}\prod_{k=1}^{K} q(z_i^k|f))(\prod_{k}^{K} q(\vartheta_k)), \tag{7.1}$$

where $q(\vartheta) \triangleq \delta(\theta_k)$, δ being the Dirac distribution, and $q(z_i^k|f) = g_\phi(f)_i^k$. g_ϕ is our network model taking as input the optical flow volume and returning a probabilistic motion segmentation volume.

We can also write the data-likelihood over a dataset $\mathcal{D}$ of optical flow volumes f as:

$$\log(\mathcal{D}) = \sum_{f\in D}\log p(f) = \sum_{f\in D}(\mathbb{E}_{q(z,\vartheta|f)}[\log\frac{p(f,z,\vartheta)}{q(z,\vartheta|f)}] + KL(q(z,\vartheta|f)||p(z,\vartheta|f))]). \tag{7.2}$$

By developing the first term and the second term being positive, we get the Evidence

Lower Bound (ELBO):

$$\log(\mathcal{D}) = \sum_{f \in D} \log p(f) \geq \sum_{f \in D} (\mathbb{E}_{q(z,\vartheta|f)}[\log p(f|z,\vartheta)] - KL(q(z,\vartheta|f)||p(z,\vartheta))). \tag{7.3}$$

Likelihood

Following the assumption stated above, we can write our likelihood as:

$$p(f|z,\vartheta) = \prod_{i \in \Omega} \prod_{k=1}^{K} p(f(i)|\vartheta_k, z) \propto \prod_{i \in \Omega} \prod_{k=1}^{K} \exp(-\frac{1}{\alpha}\delta(f(i), \tilde{f}_{\vartheta_k}(i)))^{z_i^k}, \tag{7.4}$$

where $\tilde{f}_{\vartheta_k}(i)$ is the flow vector given by the parametric motion model of parameters ϑ_k at point i. Here $\delta(.,.)$ denotes a distance between the two terms. From this likelihood, we can infer the reconstruction loss :

$$\begin{aligned}
\mathcal{L}_r = -\mathbb{E}_{q(z,\vartheta|f)}[\log p(f|z,\vartheta)] &= -\sum_{i \in \Omega}\sum_{k} \mathbb{E}_q[\log p(f(i)|\vartheta_k, z)] \\
&= -\sum_{i \in \Omega}\sum_{k} \mathbb{E}_q[z_i^k]\, \mathbb{E}_q[-\frac{1}{\alpha}(\delta(f(i), \tilde{f}_{\vartheta_k}(i))] \\
&= \sum_{i \in \Omega}\sum_{k} \frac{1}{\alpha} g_\phi(f)_i^k \delta(f(i), \tilde{f}_{\theta_k})
\end{aligned} \tag{7.5}$$

Prior

The term $KL(q(z,\vartheta|f)||p(z,\vartheta)))$ allows us to define a prior over the segmentation. In our case, we enforce the prior that every mask have an equal probability, thus $p(z_i^k) = \frac{1}{K}$. We do not employ a prior on ϑ.

$$KL(q(z)||p(z)) = \sum_{i \in \Omega} KL(q(z_i)||p(z_i)) \tag{7.6}$$

$$= \sum_{i \in \Omega}\sum_{k=1}^{K} [q(z_i = k)\log(q(z_i = k)) - q(z_i = k)\log(K)] \tag{7.7}$$

$$= \sum_{i \in \Omega}\sum_{k=1}^{K} [q(z_i = k)\log(q(z_i = k))] - |\Omega|\log(K) \tag{7.8}$$

$$= \sum_{i \in \Omega}\sum_{k=1}^{K} g_\phi(f)_i^k \log g_\phi(f)_i^k + cst. \tag{7.9}$$

Training

The loss function is thus defined by:

$$\mathcal{L} = \sum_{f \in D} - \mathbb{E}_{q(z,\vartheta|f)}[\log p(f|z,\vartheta)] + KL(q(z,\vartheta|f)||p(z,\vartheta)) \tag{7.10}$$

$$= \sum_{f \in D} \frac{1}{\alpha} \sum_{i \in \Omega} \sum_{k} g_\phi(f)_i^k \delta(f(i), \tilde{f}_{\theta_k}(i)) + g_\phi(f)_i^k \log g_\phi(f)_i^k \tag{7.11}$$

We get the same loss as in eq.2.15.

7.1.2 Data augmentation

In this section, we formally analyze the impact of the data augmentation on the motion segmentation.

Theorem 1. *Considering a function ll such as:*

$$ll(\theta, m, f) = \sum_i \sum_k m_i^k (\log p(f_i, z_i^k|\theta_k) - \log m_i^k), \tag{7.12}$$

where $\theta \in \mathbb{R}^{12}$ *are the parameters of a parametric (quadratic) motion model,* $m \in [0,1]^{W \times H \times K}$ *is a segmentation mask and* $f \in \mathbb{R}^{W \times H \times 2}$ *is the input optical flow. We define* $f_\zeta \in \mathbb{R}^{W \times H \times 2}$ *as a parametric flow field generated using parameters* θ_ζ *and positions* $c(i)$ *as* $f_{\zeta,i} = \theta_\zeta \cdot c(i)$*. We show that for all possible segmentation* m *and parameters* θ_ζ *we have :*

$$\max_\theta ll(\theta, m, f) = \max_\theta ll(\theta, m, f + f_\zeta). \tag{7.13}$$

Remark. In this theorem, *ll* and $c(i)$ are the lower bound and the polynomial terms, respectively, both of which are described in Chapter 2. For the lower bound, we take m instead of $g_\phi(f)$, since we are not specifically considering the network here. The goal here is to show that the lower bound we optimize in Chapter 2 is invariant to the added parametric global flow with respect to the segmentation. Thus, a segmentation will correspond to the same loss value, regardless of the perturbation of the input flow through the data augmentation, which encourages the network to produce global-motion-invariant segmentations. We can see an illustration of this property in Fig.7.1. Furthermore, a direct consequence of this property is that the optimal m^* segmentation is the same for

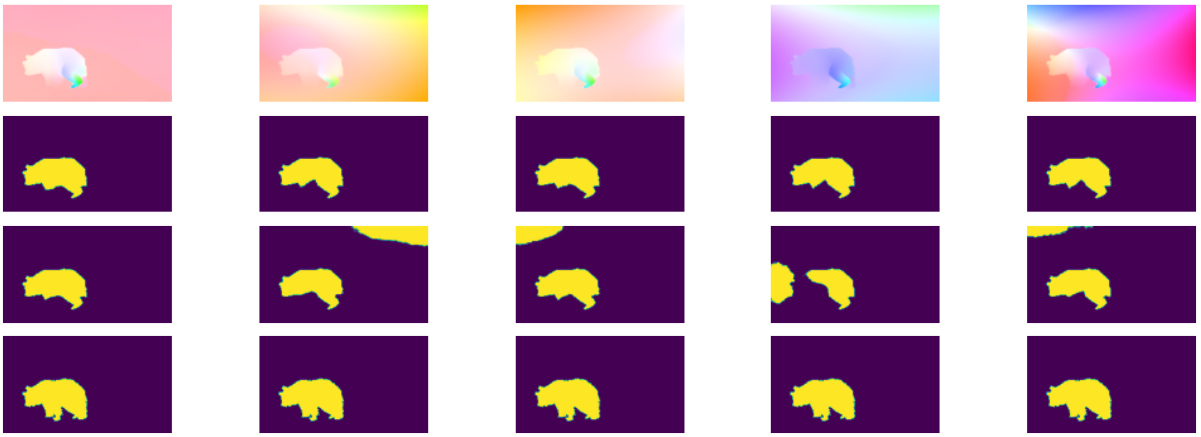


Figure 7.1 – Data augmentation adding a quadratic global motion on the optical flow field. From top to bottom : Optical flow field displayed with the usual HSV code [59]; Predicted mask with our network trained using data augmentation; Predicted mask with our network trained without data augmentation; Ground-truth segmentation mask.

both the original and augmented flows:

$$m^* = \arg\max_m [\max_\theta ll(\theta, m, f)] = \arg\max_m [\max_\theta ll(\theta, m, f + f_\zeta)]. \tag{7.14}$$

Proof. Starting from eq.4 and using eq.10 for the likelihood definition in Chapter 2, we can write:

$$\begin{aligned} ll(\theta, m, f) &= \sum_i \sum_k m_i^k (\log p(f_i, z_i^k | \theta_k) - \log m_i^k) && (7.15)\\ &= \sum_i \sum_k m_i^k \log p(f_i, z_i^k | \theta_k) - m_i^k \log m_i^k && (7.16)\\ &= \sum_i \sum_k m_i^k \log p(f_i | z_i^k, \theta_k) + m_i^k \log \left(\frac{p(z_i^k)}{p(m_i^k)} \right) && (7.17)\\ &= -\sum_i \sum_k m_i^k \delta(f_i, \theta_k^T \cdot c(i)) + \kappa, && (7.18) \end{aligned}$$

where $\kappa \triangleq m_i^k \log \left(\frac{p(z_i^k)}{p(m_i^k) Z} \right)$ is independent of f and θ. See Theorem 2 justifying this statement for the normalisation factor Z. Consequently, we have:

$$ll(\theta, m, f + f_\zeta) = -\sum_i \sum_k m_i^k \delta(f_i + f_{\zeta,i}, \theta_k^T \cdot c(i)) + \kappa. \tag{7.19}$$

Using translation invariance of our distance function δ and the additivity property of our parametric motion models (that are linear with respect to the parameters), we can write:

$$\delta(f_i + f_{\zeta,i}, \theta_k^T \cdot c(i)) = \delta(f_i, \theta_k^T \cdot c(i) - f_{\zeta,i}) = \delta(f_i, \theta_k^T \cdot c(i) - \theta_\zeta^T \cdot c(i)) \tag{7.20}$$

$$= \delta(f_i, (\theta_k - \theta_\zeta)^T \cdot c(i)) = \delta(f_i, \tilde{\theta}_k^T \cdot c(i)). \tag{7.21}$$

Thus, we have:

$$\max_{\tilde{\theta}} ll(\tilde{\theta}, m, f + f_\zeta) = \max_{\tilde{\theta}} -\sum_i \sum_k m_i^k \delta(f_i, \tilde{\theta}_k^T \cdot c(i)) + \kappa. \tag{7.22}$$

With a change of variable, we get:

$$\max_{\theta} ll(\theta, m, f) = \max_{\theta} ll(\theta, m, f + f_\zeta). \tag{7.23}$$

□

7.1.3 Normalisation factor

Theorem 2. *Let a translationally invariant function* $\delta : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ *such that,* $\forall\ a, e, c \in \mathbb{R}^D \times \mathbb{R}^D,\ \delta(a + c, e + c) = \delta(a, e)$*, and a probability density defined as:*

$$p(y|x;\theta) = \frac{1}{Z} \exp(-\delta(y, \theta^T x)). \tag{7.24}$$

Then, the normalisation factor Z *is independent of* θ*.*

In particular, this result is true for the p-norm defined by:

$$\delta(a, b) = ||a - b||_p = \Big(\sum_i |a_i - b_i|^p \Big)^{1/p}. \tag{7.25}$$

Proof. The normalisation factor is defined by:

$$Z \triangleq \int_{\mathbb{R}^{\mathbb{D}}} \exp(-\delta(y, \theta^T x)) \mathrm{d}y. \tag{7.26}$$

Taking $b \triangleq y - \theta^T x$, we get:

$$Z = \int_{\mathbb{R}^{\mathbb{D}}} \exp(-\delta(b + \theta^T x, \theta^T x)) \mathrm{d}b = \int_{\mathbb{R}^{\mathbb{D}}} \exp(-\delta(b, 0)) \mathrm{d}b. \tag{7.27}$$

Thus, Z is independent of θ, x and y, and only depends on function δ.
For the p-norm:

$$\delta(a+c, e+c) = ||a+c-e-c||_p = ||a-e||_p = \delta(a,e). \tag{7.28}$$

□

7.1.4 Algorithm

```
def TrainingStep(Flow, ConvNet, alpha=0.01, learning_rate=0.01) :
    Segmentation = SoftMax(ConvNet(Flow)) # Extract Segmentation Probabilistic Mask using any Convolutional Net
    Theta = ComputeThetaOptim(Flow, Segmentation) # Compute theta for each segment minimizing ll
    Theta = Theta.detach() # Stop Gradient on theta estimation ( alternate optimisation )

    ll = CoherenceLoss(Theta) + alpha*Entropy(Segmentation) # Equation in Section 4.2
    ConvNet = OneStepOptimiseNetwork(ll, ConvNet, learning_rate) # One step of Adam on network's weights.

def ComputeThetaOptim(Flow, Segmentation, delta='L1_norm') :
    for k in {1..K} : # For each motion segment k
            ll_k = Segmentation_k*delta(Flow, Theta_k . c(i)) # Coherence loss associate to the segment
        Theta_k = Minimize(ll_k) # LBGFS to minimize ll_k
    return Theta

def InferenceStep(Flow, ConvNet) :
    return ArgMax(ConvNet(Flow)) # For each site return the most likely mask
```

Figure 7.2 – Pseudo-Code of the training and inference steps of our method. Let us emphasize that each training step and the inference step are not iterative.

Code available at : https://github.com/Etienne-Meunier/EM-Flow-Segmentation

7.1.5 Failure cases on SegTrackV2

We observed that the optical flow is not well estimated by the RAFT method in several frames of SegTrackV2 dataset, causing our algorithm to fail on those frames. These failure cases are reported in Fig.7.3.

7.1.6 Detailed results per sequences of the datasets

Hereafter, we report detailed results through tables collecting the evaluation scores obtained by our EM method for every sequence of the four datasets, DAVIS2016, SegTrackV2, FBMS59 and MoCA.

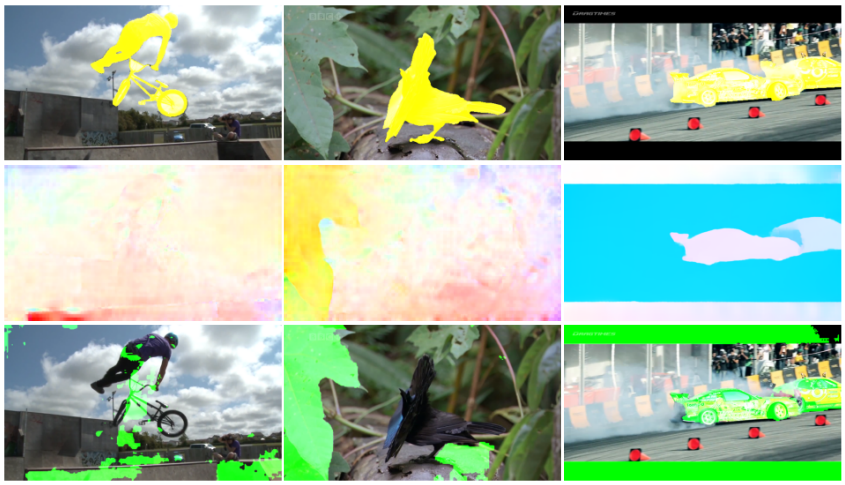


Figure 7.3 – Examples of motion segmentation results obtained with our method with two masks, on the bmx, birds of paradise and drift videos of the SegTrackV2 dataset. First row: a frame of the video with the ground-truth superimposed in yellow. Second row: the input flow field displayed with the HSV color code. Third row: the segmentation produced by our method superimposed in green on the corresponding image.

DAVIS2016

Sequence	$\mathcal{J}$ (M)	$\mathcal{J}$ (O)	$\mathcal{J}$ (D)	$\mathcal{F}$ (M)	$\mathcal{F}$ (O)	$\mathcal{F}$ (D)
blackswan	0.499	0.5	-0.091	0.608	1	-0.022
bmx-trees	0.602	0.782	0.241	0.789	0.923	0.142
breakdance	0.754	0.939	-0.013	0.789	1	0.012
camel	0.827	1	0.139	0.793	1	0.146
car-roundabout	0.908	1	-0.035	0.804	1	-0.067
car-shadow	0.89	1	0.042	0.83	1	0.044
cows	0.856	1	0.031	0.773	1	0.032
dance-twirl	0.8	1	-0.087	0.831	1	-0.024
dog	0.808	1	-0.09	0.743	0.983	-0.085
drift-chicane	0.674	0.78	-0.193	0.755	0.86	-0.065
drift-straight	0.896	1	0.04	0.855	1	0.217
goat	0.134	0	0.099	0.408	0.25	-0.049
horsejump-high	0.805	1	0.071	0.873	1	-0.001
kite-surf	0.427	0.271	0.219	0.483	0.396	0.004
libby	0.461	0.553	0.47	0.665	0.766	0.297
motocross-jump	0.651	0.737	0.037	0.576	0.658	0.093
paragliding-launch	0.623	0.667	0.308	0.327	0.205	0.406
parkour	0.563	0.551	-0.014	0.701	0.847	0.118
scooter-black	0.784	1	-0.262	0.669	1	-0.099
soapbox	0.891	1	0.015	0.865	1	0.008
Average	0.693	0.789	0.046	0.707	0.844	0.055

Table 7.1 – Results given for every sequence of DAVIS2016 dataset. Reported score is the average Jaccard score over frames in the sequence. Last row is the average over sequences scores. $\mathcal{J}$ is the Jaccard index and $\mathcal{F}$ is the Countour Accuracy. The Mean (M) is the average of the score, the Recall (O) is the fraction of frames with a score higher than 0.5 and the Decay (D) is the degradation of the score over time in the sequence. More details in [15].

SegTrackV2

Sequence	Jacc ($\mathcal{J}$)
bird of paradise	54.4
birdfall	40.7
bmx	69.7
cheetah	39.7
drift	36.1
frog	77.6
girl	64.0
hummingbird	65.5
monkey	62.0
monkeydog	12.6
parachute	92.7
penguin	32.6
soldier	72.9
worm	41.8
Seq. Avg.	54.5
Frames. Avg.	55.5

Table 7.2 – Results given for every sequence of SegTrackV2 dataset. Reported score is the average Jaccard score over annotated frames in the sequence.

FBMS59

Sequence	Jacc ($\mathcal{J}$)
camel01	65.0
cars1	87.1
cars10	33.2
cars4	86.6
cars5	83.7
cats01	68.7
cats03	78.3
cats06	39.4
dogs01	72.9
dogs02	69.8
farm01	81.0
giraffes01	34.2
goats01	43.7
horses02	75.9
horses04	67.8
horses05	43.0
lion01	56.1
marple12	50.0
marple2	70.4
marple4	80.6
marple6	35.5
marple7	62.4
marple9	31.0
people03	53.3
people1	78.7
people2	86.1
rabbits02	44.6
rabbits03	40.2
rabbits04	42.9
tennis	72.3
Seq. Avg.	61.1
Frames. Avg.	57.8

Table 7.3 – Results given for every sequence of FBMS59 dataset. Reported score is the average Jaccard score over annotated frames in the sequence.

MoCA

Table 7.4 – Results given for every sequence of MoCA dataset. Reported score is the average Jaccard score over frames in the sequence computed using bounding box annotation as described in Section 5.2. of the main text.

Sequence	Jacc ($\mathcal{J}$)
arabian horn viper	71.0
arctic fox	37.8
arctic fox 1	85.5
arctic wolf 0	63.4
arctic wolf 1	50.8
bear	61.3
black cat 0	53.5
black cat 1	8.6
crab	68.1
crab 1	29.3
cuttlefish 0	23.5
cuttlefish 1	20.0
cuttlefish 4	64.5
cuttlefish 5	72.1
dead leaf butterfly 1	75.7
desert fox	27.0
devil scorpionfish	90.3
devil scorpionfish 1	92.5
devil scorpionfish 2	85.7
egyptian nightjar	72.4
elephant	80.4
flatfish 0	59.9
flatfish 1	64.8
flatfish 2	82.1
flatfish 4	22.1
flounder	89.3
flounder 3	21.4

flounder 4	77.7
flounder 5	69.7
flounder 6	80.9
flounder 7	51.5
flounder 8	69.7
flounder 9	71.6
fossa	16.6
goat 0	66.6
goat 1	72.2
groundhog	56.5
hedgehog 0	43.4
hedgehog 1	55.6
hedgehog 2	69.9
hedgehog 3	50.9
hermit crab	68.8
ibex	25.8
jerboa	54.8
jerboa 1	41.7
lichen katydid	67.8
lion cub 0	80.1
lion cub 1	54.2
lion cub 3	23.0
lioness	25.9
marine iguana	44.8
markhor	80.0
meerkat	88.6
mountain goat	73.9
nile monitor 1	38.6
octopus	63.6
octopus 1	55.4
peacock flounder 0	94.6
peacock flounder 1	86.1

peacock flounder 2	89.4
polar bear 0	0.0
polar bear 1	21.7
polar bear 2	75.5
pygmy seahorse 2	47.0
pygmy seahorse 4	68.4
rodent x	63.4
scorpionfish 0	67.2
scorpionfish 1	63.7
scorpionfish 2	85.9
scorpionfish 3	82.8
scorpionfish 4	84.1
scorpionfish 5	85.7
seal 1	86.2
seal 2	56.2
seal 3	51.3
shrimp	69.4
snow leopard 0	75.6
snow leopard 1	82.7
snow leopard 2	90.9
snow leopard 3	69.7
snow leopard 6	87.8
snow leopard 7	68.1
snow leopard 8	61.3
snowy owl 0	59.8
spider tailed horned viper 0	40.7
spider tailed horned viper 1	52.5
spider tailed horned viper 2	81.4
spider tailed horned viper 3	83.3
Seq. Avg.	61.9
Frames. Avg.	61.8

7.2 Appendix B: Short-term Motion Segmentation

7.2.1 Repeatability

In order to evaluate the reliability of our method, we repeated five times the training of our model with five different initialisations and performed the abovementioned evaluation pipeline.

Experiment	DAVIS Val	SegTrackV2	FBMS
1	73.2	55.0	59.4
2	73.9	57.6	59.0
3	72.6	55.1	59.5
4	73.9	56.5	60.5
5	72.7	55.2	59.1
Average	73.3	55.9	59.5

Table 7.5 – Results (Jaccard index) of five experiments, involving different initialisations of the network, on the three datasets. Reported results in Chapter 3 correspond to experiment 1.

We can observe that the results collected in Table 7.5 are stable, whereas the process described above (segment linkage and segment selection) could generate variability.

7.2.2 Additional experiments

Training without data augmentation

To extend our ablation study, we have also carried out the evaluation of the case when our network is trained without any data augmentation. Results are collected in Table 7.6. Clearly, as expected, the data augmentation improves performance.

Data Augmentation	Davis Val	SegTrackV2	FBMS
With	73.2	55.0	59.4
Without	70.1	52.3	50.4

Table 7.6 – Impact of the data augmentation. Scores (Jaccard index) obtained on the three datasets.

Impact of the temporal interval τ

Regarding the τ parameter (i.e., the temporal interval between the flows of the input triplet), we trained our network with the full configuration and randomly sampled τ values at training time. Let us remind that the flow $f_{t-\tau}$ (respectively, $f_{t+\tau}$) is computed between image frames at time instants $t-\tau$ and $t-\tau+1$ (respectively, $t+\tau$ and $t+\tau+1$). We uniformly sample τ among a set of values during training. At inference, we still use only $\tau = 1$ for the sake of efficiency. Thus, this experiment could also be perceived as a type of data augmentation.

τ while training	Davis Val	SegTrackV2	FBMS
{1}	73.2	55.0	59.4
{1,2,3,4,5,9,10,12}	73.0	54.3	57.9

Table 7.7 – Impact of the use, at training time, of different values for the time interval τ in the input flow triplet. Scores (Jaccard index) obtained on the three datasets.

As we can observe in Table 7.7, it has no sensible impact on results (even a slight performance decrease). In future work, we plan to investigate the combination of different τ values, including negative values, at test time.

7.2.3 Latent motion representation

We give a few highlights on the latent motion representation issued from the trained network as mentioned in Chapter 3. We carried out a preliminary experiment directly based on the normalized latent vectors of all the sites of a subsequence. The latent vectors are of dimension 32. We applied a PCA procedure to all the latent vectors over the whole subsequence of length T and taking into account the triplets at each time instant. These latent vectors are stacked as an array of dimensions $(3 \times H \times W \times T, 32)$, where H and W are respectively the height and the width of each frame of the subsequence.

Then, we compute the softmax of the projections onto the three first components of the PCA output. Interestingly, after thresholding the softmax values (threshold value of 0.7), we observe that the resulting map is likely to provide a binary segment close to the ground truth of the primary moving object, as illustrated in Fig.7.4. It shows that our latent motion representation is not only informative in its own, but more importantly, is coherent over the subsequence since the PCA is computed once over the subsequence. In future work, we will investigate further this possibility to provide a binary segmentation

directly oriented to the VOS evaluation, when our network is trained for multiple motion segmentation with K masks.

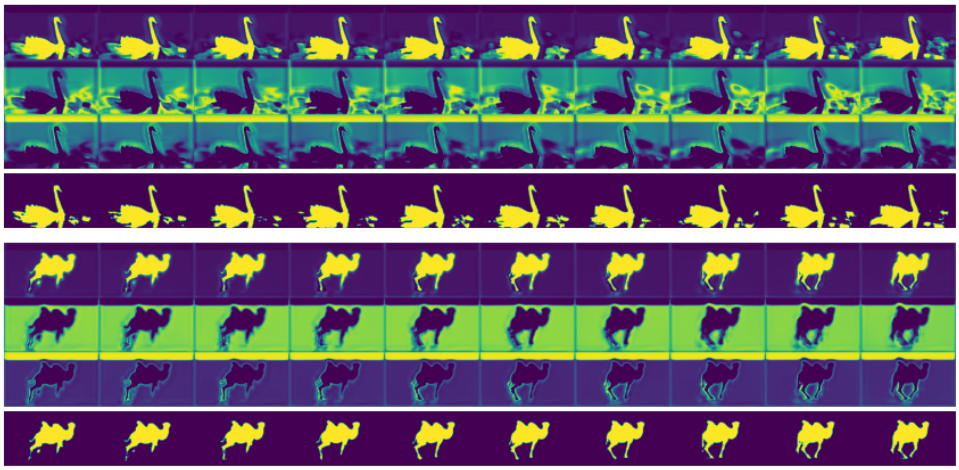


Figure 7.4 – Illustration of the principal component analysis of the latent motion representation. Two examples from DAVIS2016: *blackswan* and *camel* videos. For each example, the first three rows are the projection of the latent vectors on the three first principal components. The fourth row is the binary segmentation obtained by thresholding the projection on the first component.

7.2.4 Detailed results per videos of the datasets

Hereafter, we report detailed results through tables collecting the evaluation scores obtained by our ST-MS method for every video of the four datasets, DAVIS2016, SegTrackV2, FBMS59, and DAVIS2017-motion. Let us recall that the official evaluation algorithm is not the same for DAVIS2016 and DAVIS2017. The evaluation is done in one go on the whole video for DAVIS2017, while it is achieved frame by frame of the video for DAVIS2016.

DAVIS2016

Video	$\mathcal{J}$ (M)	$\mathcal{J}$ (O)	$\mathcal{J}$ (D)	$\mathcal{F}$ (M)	$\mathcal{F}$ (O)	$\mathcal{F}$ (D)
blackswan	0.584	0.833	-0.11	0.594	0.875	-0.072
bmx-trees	0.597	0.756	0.198	0.798	0.949	0.095
breakdance	0.738	0.976	0.004	0.738	0.988	-0.007
camel	0.871	1	0.12	0.859	1	0.128
car-roundabout	0.918	1	-0.026	0.828	1	-0.068
car-shadow	0.897	1	0.019	0.846	1	-0.011
cows	0.873	1	0.031	0.804	1	0.022
dance-twirl	0.821	1	-0.071	0.853	1	-0.022
dog	0.812	1	-0.044	0.709	0.931	-0.024
drift-chicane	0.664	0.8	-0.221	0.754	0.84	-0.069
drift-straight	0.861	1	0.046	0.786	0.938	0.245
goat	0.298	0	0.125	0.299	0.023	0.037
horsejump-high	0.821	1	0.097	0.87	1	0.044
kite-surf	0.424	0.354	0.249	0.41	0.208	0.087
libby	0.734	0.979	0.117	0.846	1	-0.002
motocross-jump	0.61	0.553	0.182	0.377	0.474	0.198
paragliding-launch	0.624	0.667	0.313	0.314	0.167	0.375
parkour	0.735	0.959	0.069	0.777	1	0.15
scooter-black	0.861	1	-0.023	0.739	1	0.106
soapbox	0.889	1	0.03	0.859	1	0.021
Average	0.732	0.844	0.055	0.703	0.82	0.062

Table 7.8 – Results given for every video of DAVIS2016 dataset. Reported scores per video are the average Jaccard score over frames in the video. The very last row is the average over videos scores. $\mathcal{J}$ is the Jaccard index and $\mathcal{F}$ is the Countour Accuracy. The Mean (M) is the average of the scores, the Recall (O) is the fraction of frames per video with a score higher than 0.5, and the Decay (D) is the degradation of the score over time in the video.

SegTrackV2

Video	Jacc ($\mathcal{J}$)
Bird of paradise	51.5
birdfall	38.1
bmx	76.9
cheetah	44.1
drift	33.0
frog	78.2
girl	59.8
hummingbird	68.7
monkey	53.9
monkeydog	16.6
parachute	92.9
penguin	39.4
soldier	64.0
worm	39.3
Frames. Avg	55.0

Table 7.9 – Results given for every video of SegTrackV2 dataset. Each reported score is the average Jaccard score over annotated frames in the video. The very last row is the average over all the frames and over all the videos.

FBMS59

Video	Jacc ($\mathcal{J}$)
camel01	27.8
cars1	88.1
cars10	54.1
cars4	83.6
cars5	83.7
cats01	71.1
cats03	79.9
cats06	38.9
dogs01	73.1
dogs02	66.1
farm01	79.1
giraffes01	36.2
goats01	45.5
horses02	65.3
horses04	72.4
horses05	43.9
lion01	50.7
marple12	61.3
marple2	64.3
marple4	77.8
marple6	51.0
marple7	58.0
marple9	66.8
people03	52.8
people1	80.1
people2	87.3
rabbits02	49.8
rabbits03	41.3
rabbits04	50.2
tennis	72.6
Frames. Avg.	59.4

Table 7.10 – Results given for every video of FBMS59 dataset. Each reported score is the average Jaccard score over annotated frames in the video. The very last row is the average over all the annotated frames and over all the videos.

DAVIS2017-motion

Sequence	J-Mean	F-Mean
bike-packing_1	0.072	0.370
bike-packing_2	0.276	0.393
blackswan_1	0.577	0.593
bmx-trees_1	0.520	0.766
breakdance_1	0.365	0.558
camel_1	0.716	0.683
car-roundabout_1	0.900	0.814
car-shadow_1	0.870	0.804
cows_1	0.778	0.675
dance-twirl_1	0.441	0.641
dog_1	0.481	0.456
dogs-jump_1	0.433	0.506
dogs-jump_2	0.018	0.174
dogs-jump_3	0.190	0.251
drift-chicane_1	0.381	0.562
drift-straight_1	0.811	0.739
goat_1	0.275	0.303
gold-fish_1	0.175	0.270
gold-fish_2	0.027	0.325
gold-fish_3	0.168	0.209
gold-fish_4	0.000	0.000
gold-fish_5	0.000	0.000
horsejump-high_1	0.562	0.783
india_1	0.057	0.066
india_2	0.047	0.110
india_3	0.143	0.186
judo_1	0.247	0.357
judo_2	0.417	0.522
kite-surf_1	0.422	0.418
lab-coat_1	0.337	0.313
libby_1	0.730	0.847
loading_1	0.166	0.226
loading_2	0.068	0.210
loading_3	0.407	0.416
mbike-trick_1	0.644	0.683
motocross-jump_1	0.509	0.481
paragliding-launch_1	0.339	0.237
parkour_1	0.682	0.753
pigs_1	0.253	0.394
pigs_2	0.019	0.313
pigs_3	0.369	0.432
scooter-black_1	0.856	0.750
shooting_1	0.575	0.500
soapbox_1	0.726	0.779

J&FMean	JMean	JRecall	JDecay	FMean	FRecall	FDecay
0.420	0.388	0.365	0.006	0.452	0.454	0.039

Table 7.11 – Results given for every video of DAVIS2017-motion dataset. The very last row is the average score over all the videos for the different criteria.

7.3 Appendix C: Long-term Motion Segmentation

7.3.1 Additional ablation study

Training without data augmentation

We also conducted an ablation study regarding data augmentation. Results are collected in Table 7.12. A similar conclusion can be established as the one drawn for the ablation study applied to the three main components of our LT-MS method in Chapter 4, yet with a larger margin in this case.

Dataset	DAVIS2016		FBMS59		SegTrackV2	
Cut Size	10	120	10	120	10	120
Full Model	74.8	72.4	61.0	58.2	61.3	60.4
No Data Augmentation	72.6	69.4	55.8	51.6	56.4	54.4

Table 7.12 – Ablation study for the data augmentation component of our method LT-MS ($K = 4$) on the three datasets DAVIS2016, FBMS59 and SegTrackV2. The performance scores are given by the Jaccard index $\mathcal{J}$; the higher $\mathcal{J}$, the better. We report ablation results with two input-flow sequence lengths (or cut size), respectively, by dividing the video into pieces of ten successive frames or by considering 120 successive frames (in practice, the whole video for the DAVIS2016 dataset).

7.3.2 Repeatability

With the introduction of the transformer decoder, we experimentally found that the convergence of the network depends on the weights initialization, and that the same network and loss configuration can yield different results at test time. In Fig.7.5, we show that our unsupervised loss on a held-out validation set is a good indicator of the network performance at test time. This is a critical point for model and hyperparameter selection, since we do not have access to the ground truth at training time (with our fully unsupervised scenario), and thus we cannot evaluate model performance. Fig.7.6 plots the model performance after training as a function of the initialization budget.

In our evaluation experiments, we account for this randomness by training five models with the same set of seeds for all ablations and by reporting the score of the model with the lowest validation loss for each model.

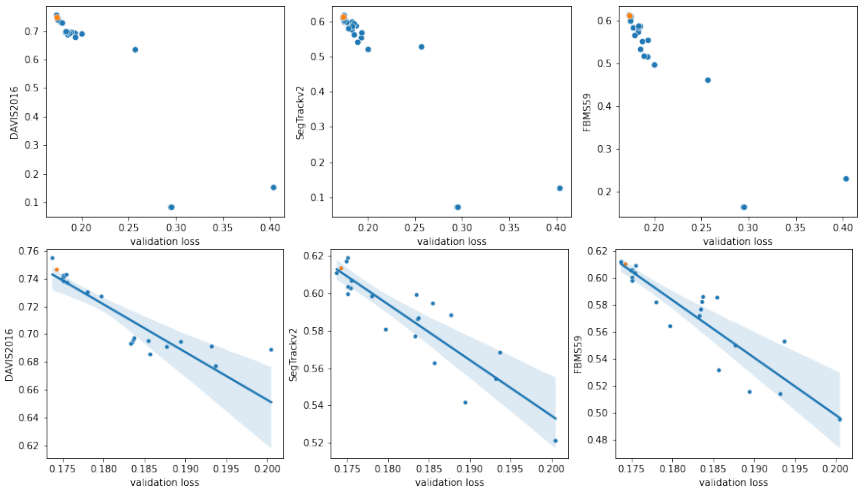


Figure 7.5 – Model scores depending on initialization. Each dot represents a trained model. The x-axis represents the validation loss on the DAVIS 2016 train dataset, and the y-axis the performance on the evaluation dataset. The top row includes all models and the bottom row excludes models that diverged during training (validation loss > 0.25). The bottom row displays the linear relationship between validation loss and evaluation score. The model whose results are reported in the main text is represented as an orange dot.

7.3.3 Addition of Slot Attention

Several works on video object segmentation [46], [49] include the slot attention mechanism [120]. This attention mechanism takes place in the transformer decoder and starts with a random set of queries that are then iteratively refined depending on the input data, similar to a classical soft k-means algorithm, we illustrate this process with Figure 7.7.

This is an interesting addition, as it theoretically allows you to vary the number of output masks without having to retrain the network, as highlighted in [49]. In practice, this is only true if the initial query vector is sampled randomly, as in the original paper, which is done in [49]. However, in [46], the authors mention that they were unable to train a model with Gaussian initial query vectors, and so they used learnable queries, which makes their method not adaptable to a variable number of outputs, despite the fact that they use slot attention.

To compare with the classical transformer decoder and to develop a method with a variable number of masks at inference time, we tried to train our model after replacing the transformer decoder with a slot attention decoder. We call this new model LT-MS-SA. We were not able to train the network from scratch because the training diverged, we also

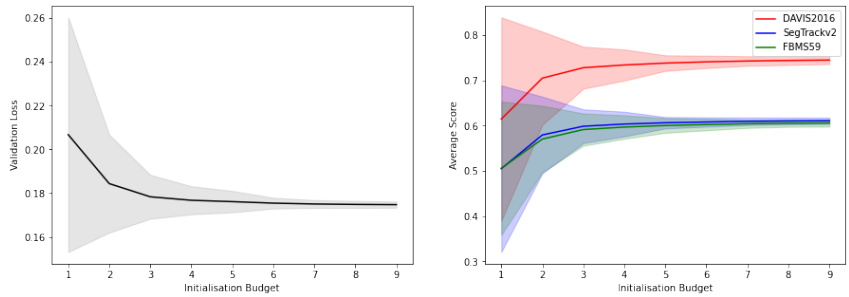


Figure 7.6 – Evolution of the model performance as a function of the initialization budget. Left : the validation loss associated to the best network of each subset. Right : average performance on the test data set associated to this network for each different budget. The filled area correspond to +/- the standard deviation for each curve.

tried to implement the "implicit differentiation" [152] version of slot attention without success. However, if we first train the U-net part of the network before training the slot attention decoder, the network converges well.

We use $K = 4$ masks at training time. At test time, we can apply our long-term motion segmentation network using different numbers K of masks.

As we can see in Table 7.13, at inference time, the predictions for the originally trained model LT-MS-SA (K=4) are on par with the model LT-MS-K4 trained with classical attention, and the LT-MS-SA model maintains good performance with $K = 3$ or $K = 5$

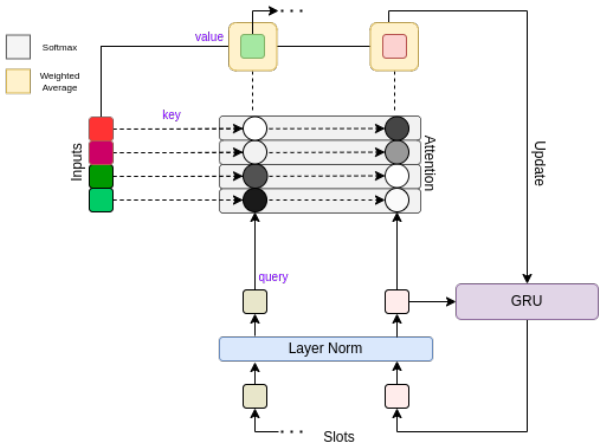


Figure 7.7 – A stage of slot attention (without the optional residual MLP). The slot representations are normalized, then their queries (slot values by query MLP) are compared to the key of the input. Updates are then computed using a weighted average of the input values. The updates are then independently passed to a GRU to obtain the new slots.

Dataset	**DAVIS2016**		**FBMS59**		**SegTrackv2**	
Cut Size	10	120	10	120	10	120
$K = 2$	54.1	50.7	46.1	43.8	40.1	39.0
$K = 3$	72.3	72.0	58.7	57.9	59.2	58.5
$K = 4$	73.4	72.7	59.5	58.4	60.1	58.9
$K = 5$	73.5	72.8	59.7	58.5	60.6	59.0
LT-MS-K2	70.3	70.7	55.3	51.9	58.6	58.9
LT-MS-K4	74.8	72.4	61.0	58.2	61.3	60.4

Table 7.13 – Results obtained with LT-MS-SA, the model which includes the slot attention decoder, for different numbers K of masks at inference. For the record, we also give the results obtained with the original LT-MS-K2 and LT-MS-K4 models without slot attention.

masks. However, the evaluation with $K = 2$ masks gives poor results compared to the model fine-tuned on two masks. A possible improvement could be to start with a LT-MS-SA model trained with $K = 3$, as the step in terms of number of masks would be smaller. We can also observe these performances in the qualitative results in Fig.7.8, where the segmentation is quite good from 3 to 6 masks and then drops when $K = 2$.

7.3.4 Detailed results per videos of the datasets

Hereafter, we report detailed results through tables collecting the evaluation scores obtained by our LT-MS-K4 method for every video of the three datasets, DAVIS2016, SegTrackV2 and FBMS59, and by our LT-MS-K3 model for every video of the DAVIS2017-motion dataset. Let us recall that the official evaluation algorithm is not the same for DAVIS2016 and DAVIS2017-motion. The evaluation is done in one go on the whole video for DAVIS2017-motion and is multi-segment, while it is binary and performed frame by frame of the video for DAVIS2016.

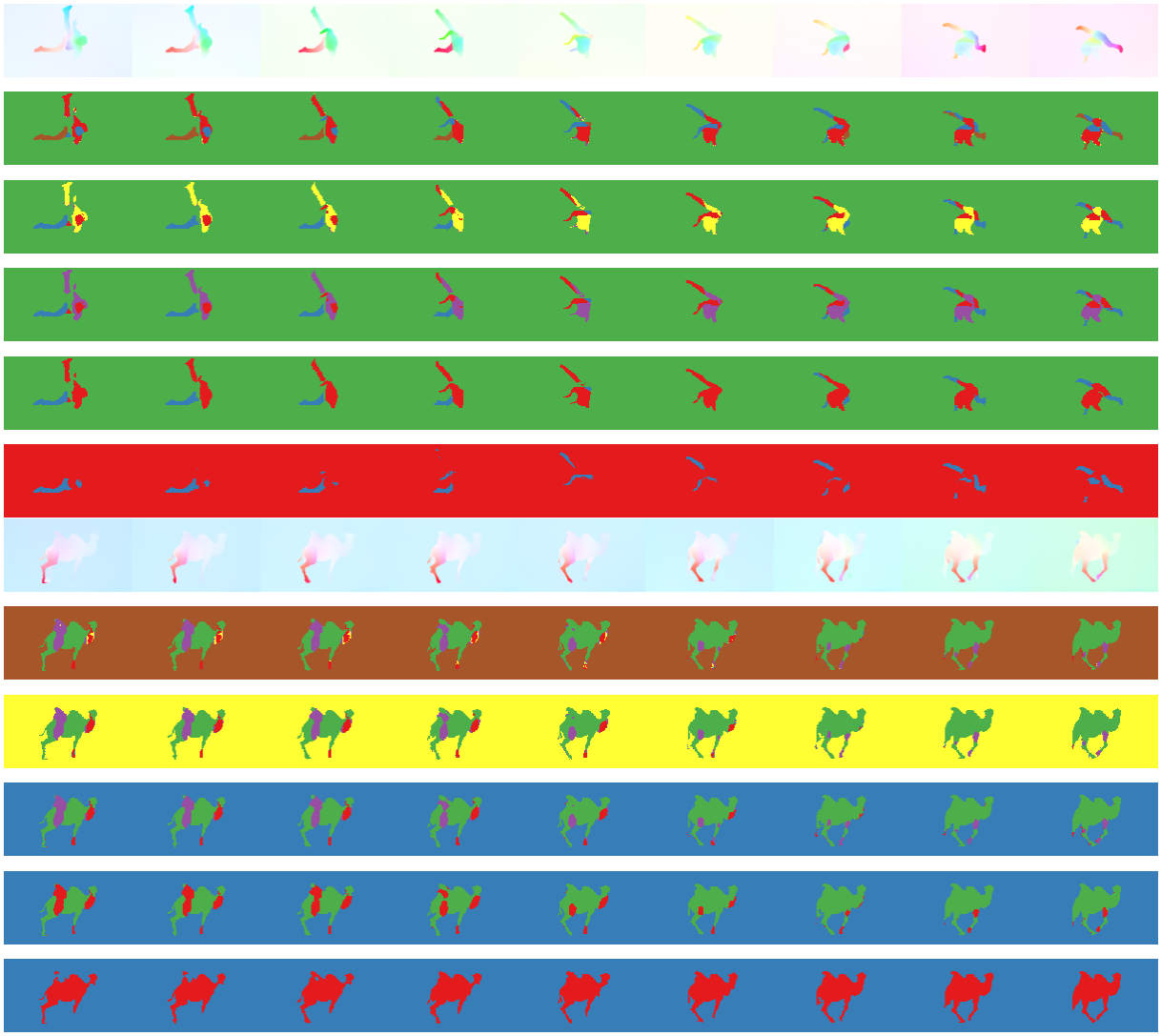


Figure 7.8 – Qualitative results obtained with LT-MS-SA (including the slot attention decoder) with a variable number of masks at inference time. The first row of the two groups of results is the optical flow field (HSV color code), then, from top to bottom, we display the motion segmentation from 6 to 2 masks using the same model. In order to push the model to keep coherent labels across levels, we keep the same set of initial queries and give only the requested number for each level as input. The two sequences are *breakdance-flare* and *camel* from the DAVIS2016 dataset.

DAVIS2016

Video	$\mathcal{J}$ (M)	$\mathcal{J}$ (O)	$\mathcal{J}$ (D)	$\mathcal{F}$ (M)	$\mathcal{F}$ (O)	$\mathcal{F}$ (D)
blackswan	0.515	0.521	-0.091	0.561	0.833	-0.036
bmx-trees	0.596	0.744	0.234	0.795	0.923	0.125
breakdance	0.740	1.000	0.038	0.722	1.000	0.000
camel	0.867	1.000	0.104	0.858	1.000	0.101
car-roundabout	0.920	1.000	-0.004	0.806	1.000	-0.060
car-shadow	0.901	1.000	0.011	0.848	1.000	-0.017
cows	0.872	1.000	0.026	0.798	1.000	0.009
dance-twirl	0.821	1.000	-0.113	0.842	1.000	-0.045
dog	0.769	1.000	-0.085	0.671	1.000	-0.076
drift-chicane	0.718	0.860	0.063	0.793	0.860	0.161
drift-straight	0.892	1.000	0.047	0.838	1.000	0.229
goat	0.380	0.330	-0.087	0.367	0.114	-0.069
horsejump-high	0.843	1.000	0.064	0.896	1.000	-0.002
kite-surf	0.518	0.500	0.114	0.494	0.375	-0.035
libby	0.781	1.000	0.108	0.892	1.000	0.035
motocross-jump	0.750	0.842	0.042	0.590	0.658	0.092
paragliding-launch	0.621	0.667	0.299	0.308	0.179	0.357
parkour	0.711	0.878	-0.315	0.765	0.949	-0.157
scooter-black	0.854	1.000	-0.036	0.729	1.000	0.072
soapbox	0.891	1.000	0.020	0.863	1.000	0.020
Average	0.748	0.867	0.022	0.722	0.845	0.035

Table 7.14 – Results given for every video of DAVIS2016 dataset. Reported scores per video are the average Jaccard score over frames in the video. The very last row is the average over videos scores. $\mathcal{J}$ is the Jaccard index and $\mathcal{F}$ is the Countour Accuracy. The Mean (M) is the average of the scores, the Recall (O) is the fraction of frames per video with a score higher than 0.5, and the Decay (D) is the degradation of the score over time in the video.

SegTrackV2

Video	Jacc ($\mathcal{J}$)
bird of paradise	0.596
birdfall	0.468
bmx	0.776
cheetah	0.438
drift	0.431
frog	0.792
girl	0.651
hummingbird	0.697
monkey	0.607
monkeydog	0.241
parachute	0.928
penguin	0.517
soldier	0.752
worm	0.514

Table 7.15 – Results given for every video of SegTrackV2 dataset. Each reported score is the average Jaccard score over annotated frames in the video. The very last row is the average over all the frames and over all the videos.

FBMS59

Video	Jacc ($\mathcal{J}$)
camel01	0.313
cars1	0.878
cars10	0.344
cars4	0.713
cars5	0.856
cats01	0.705
cats03	0.72
cats06	0.365
dogs01	0.73
dogs02	0.727
farm01	0.817
giraffes01	0.367
goats01	0.448
horses02	0.637
horses04	0.717
horses05	0.467
lion01	0.564
marple12	0.604
marple2	0.822
marple4	0.865
marple6	0.539
marple7	0.721
marple9	0.723
people03	0.587
people1	0.766
people2	0.881
rabbits02	0.498
rabbits03	0.409
rabbits04	0.483
tennis	0.714

Table 7.16 – Results given for every video of FBMS59 dataset. Each reported score is the average Jaccard score over annotated frames in the video. The very last row is the average over all the annotated frames and over all the videos.

DAVIS2017-motion

Sequence	J-Mean	F-Mean
bike-packing$_1$	0.078	0.327
bike-packing$_2$	0.254	0.267
blackswan$_1$	0.477	0.575
bmx-trees$_1$	0.608	0.804
breakdance$_1$	0.450	0.526
camel$_1$	0.772	0.726
car-roundabout$_1$	0.913	0.810
car-shadow$_1$	0.896	0.845
cows$_1$	0.821	0.728
dance-twirl$_1$	0.444	0.576
dog$_1$	0.654	0.571
dogs-jump$_1$	0.019	0.147
dogs-jump$_2$	0.254	0.323
dogs-jump$_3$	0.303	0.351
drift-chicane$_1$	0.557	0.627
drift-straight$_1$	0.886	0.829
goat$_1$	0.220	0.300
gold-fish$_1$	0.018	0.240
gold-fish$_2$	0.282	0.336
gold-fish$_3$	0.357	0.356
gold-fish$_4$	0.000	0.000
gold-fish$_5$	0.000	0.000
horsejump-high$_1$	0.721	0.794
india$_1$	0.066	0.085
india$_2$	0.163	0.172
india$_3$	0.072	0.108
judo$_1$	0.225	0.397
judo$_2$	0.286	0.414
kite-surf$_1$	0.449	0.464
lab-coat$_1$	0.332	0.350
libby$_1$	0.746	0.853
loading$_1$	0.057	0.246
loading$_2$	0.128	0.259
loading$_3$	0.427	0.514
mbike-trick$_1$	0.444	0.436
motocross-jump$_1$	0.446	0.435
paragliding-launch$_1$	0.577	0.286
parkour$_1$	0.531	0.646
pigs$_1$	0.105	0.433
pigs$_2$	0.437	0.431
pigs$_3$	0.059	0.217
scooter-black$_1$	0.843	0.724
shooting$_1$	0.434	0.568
soapbox$_1$	0.487	0.696

J&FMean	JMean	JRecall	JDecay	FMean	FRecall	FDecay
0.422	0.393	0.387	0.004	0.450	0.437	0.024

Table 7.17 – Results given for every video of DAVIS2017-motion dataset. The very last row is the average score over all the videos for the different criteria.

7.4 Appendix D: Optical flow for small moving objects in large images

7.4.1 Introduction and motivation

Appendix D describes joint work done with Sarra Khairi during her internship at Inria Rennes (March-August2023), which I co-supervised regarding methodology, programming and report writing. The internship took place in the collaboration with Airbus Defence&Space supported by the LiChIE contract (Bpifrance). The content of this section is mainly derived from her internship report that describes the work done in detail. An extended summary of the contributions and a subset of results will be given in this Appendix section.

Motion analysis in satellite image sequences leads to focus on small moving objects in large-scale images. Before attempting to segment these small moving objects, we have to make sure that we can get optical flow fields reliable and accurate enough for the segmentation task. Yet, it turned out that the RAFT method was not efficient in this specific context, and we had to look for an alternative.

Anyway, developing an optical flow (OF) method for satellite images has great potential. A similar configuration can be also encountered in live cell microscopy. A dedicated optical flow method can facilitate the interpretation of these image modalities, and helps one to build algorithms for dynamic content analysis or event detection for these applications. Several classical optical flow methods have been applied to these modalities. We could expect an improvement of the measured flow field with the introduction of deep learning, although to the best of our knowledge, no methods have been specifically developed with these modalities in mind. We will first show that state-of-the-art deep learning methods for flow estimation fail on these modalities, and then, present a new deep learning estimation method that achieves good results on satellite imagery.

RAFT [56] is one of the most popular method for optical flow computation. It has proven to deliver competitive results on many datasets. However, this method comes with several drawbacks, especially when transferred to specific applications (e.g., satellite or microscopic videos). First, this method depends on the data it was trained on. Therefore, the accuracy of flow estimations may decrease on newly seen videos whose distribution differs from the training data. Furthermore, it constructs an all-pairs 4D correlation volume, which leads to memory problems when predicting large-scale high-resolution optical

flows. Lastly, the downsampling caused by the feature extraction module misleads the flow estimation, especially for thin and small moving structures.

These limitations arise specifically when estimating optical flow on the aerial sequences provided by Airbus. These stabilized aerial image sequences mimic (future) satellite images. Indeed, making predictions on these high-resolution images using RAFT requires large memory, and the results tend to miss small moving vehicles. Also, it is not possible to finetune existing supervised methods like RAFT on aerial or satellite sequences due to the lack of video annotation and relevant synthetic data.

To illustrate our remarks, we show results of optical flow field estimation on satellite images using two existing methods in Fig.7.9. Brox method [153] is a variational method for flow estimation that appear to work well on satellite images. RAFT [56] is a commonly used deep learning method for flow estimation on classical images. As we can see, it works poorly on satellite images missing a large part of the small moving objects or merging other ones in blobs.

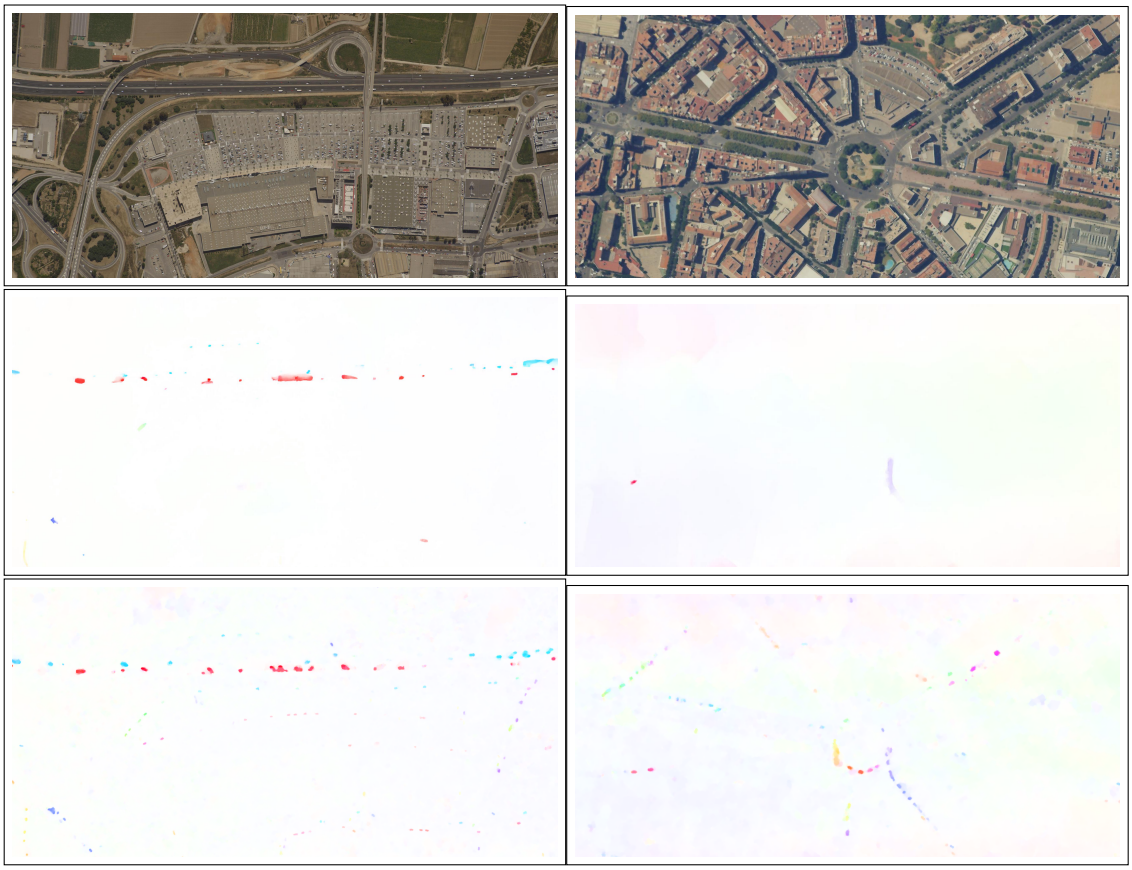


Figure 7.9 – Comparing RAFT (middle row) and Brox's method (bottom row) predictions on two sequences CenterCo (top left) and PlacaTarraco (bottom right).

7.4.2 Related work

General framework

The first end-to-end CNN-based flow estimation method dates back to FlowNet [154], a supervised neural network that either computes convolutions directly on the stacked images (FlowNetSimple) or produces meaningful representations of both images separately before comparing them at a higher level (FlowNetCorr). Once the feature extraction is done, the resulting coarse feature maps need to be upsampled back to the original resolution using strided convolutions (sometimes called *deconvolution*). This refinement then yields the final dense flow estimation. This CNN-based optical flow algorithm was trained in a supervised way on the synthetic dataset Flying Chairs that was created by the same authors.

The general framework of most existing deep flow methods can be summarized in these four fundamental steps:

1. Feature maps of both images at different resolutions are extracted;
2. Correlation is then computed in order to find correspondences between the two images;
3. Intermediate optical flow is predicted based on the correlation volume and the previous optical flow estimation;
4. Finally, the previous step (or two previous steps depending on the method) is repeated in an iterative loop.

The iterative refinement is either achieved according to a coarse-to-fine warping-based approach [98] or to a single high-resolution flow update approach [56], [155], [156].

Stacking multiple networks

FlowNet2.0 [157] proposed to improve the accuracy of optical flow estimates by stacking multiple encoder-decoder networks (FlowNetS networks) into a large model, each taking the source image and the target image warped with the previously estimated flow. Compared to the original FlowNet, FlowNet2.0 decreases the estimation error by more than 50%. However, the model is very large (160M parameters) and therefore more prone to overfitting.

Multi-resolution approach

SpyNet [155] and PWC-Net [98] embedded the traditional coarse-to-fine approach into the learning framework. This classical concept consists in estimating optical flow using a multi-scale image pyramid, from the coarsest to the finest level. SpyNet proposed a lightweight architecture with only 1.2M parameters. It showed the potential of combining CNNs with classical principles, namely pyramidal processing and warping. However, it failed to fully exploit these principles and underperformed FlowNet2.0 as well as classical energy-based methods. Another compact CNN model called PWCNet tried to solve the trade-off between model size and accuracy by using classical concepts more effectively. Instead of constructing a fixed image pyramid like SpyNet, PWC-Net designed a learnable feature pyramid by using CNNs. Besides the two classical principles listed above, PWCNet also incorporated another one: the use of cost volumes. A cost volume stores the data matching costs for the potential correspondences of each pixel. Feature pyramids from

both images are built prior to the partial cost volume computation. To generate features F_0^l and F_1^l at the l-th level, features from the lower level $l-1$ are downsampled by a factor of two using convolutional networks. It is illustrated in Fig.7.10.

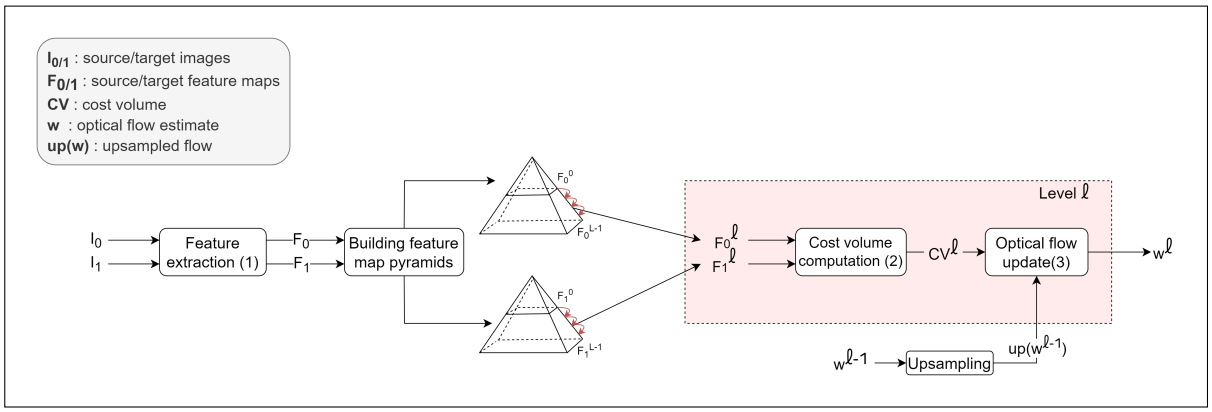


Figure 7.10 – Iterative refinement of the flow using the coarse-to-fine warping-based approach.

Single-resolution approach

The coarse-to-fine approach fails to address the well-known challenge of small, fast-moving objects. Indeed, when building the feature pyramid using successive downsampling, objects whose size is smaller than their displacement tend to disappear at the level where their flow is small enough to be estimated [158]. Furthermore, the warping process used by both PWC-Net and SpyNet can propagate early errors from higher levels, making the final optical flow estimate unreliable. RAFT overcomes these limitations by operating at a single resolution flow field. Figure 7.11 shows that features are extracted at a single resolution. At each iteration k, the flow estimate is updated by adding the generated residual flow $\Delta\mathbf{w}^k$ to the current flow estimate $\mathbf{w}^k$: $\mathbf{w}^{k+1} = \mathbf{w}^k + \Delta\mathbf{w}^k$.

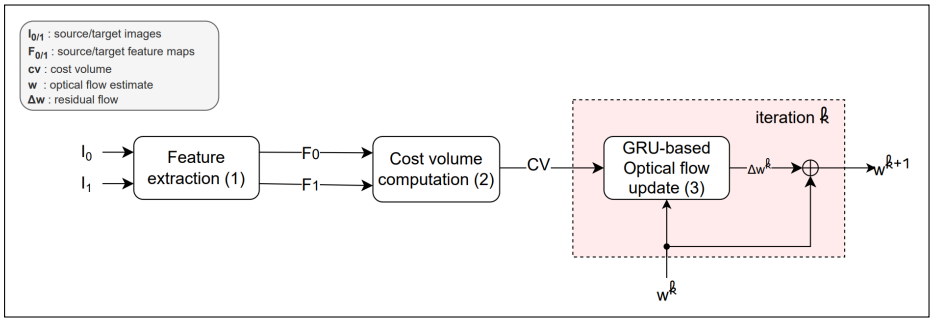


Figure 7.11 – Iterative refinement of the flow using a single high-resolution flow update

Feature extraction

RAFT and other related methods extract feature representations of input images using a CNN-based feature encoder. The feature encoder f_θ maps I_0 and I_1 to their abstract feature representations F_0 and F_1 at a lower $1/2^s$ resolution:

$$\begin{aligned} f_\theta : \ \mathbb{R}^{H\times W\times 3} &\longrightarrow \mathbb{R}^{H/2^s\times W/2^s\times C} \\ I \quad &\longmapsto \quad F, \end{aligned}$$

where θ represents the parameters of the network, C is the number of feature channels, and H and W are respectively the height and width of the images. A context encoder is generally added to extract features from the source image. The authors of RAFT showed that injecting the context into the update block improves optical flow results. The intuition behind this is that it aggregates spatial information with motion boundaries.

Computing visual similarity

Once the features are extracted, the next task is to build the correlation volume that stores the visual similarities between pixels in the source and target frames. There are various ways to formulate the computation of the cost volume.

The first one is to adopt a local correlation volume. To perform the matching process, FlowNetCorr introduced a local correlation volume that can be expressed as follows:

$$\mathrm{CV} = \Big\{ F_0(\mathbf{p}) \cdot F_1(\mathbf{p}+\mathbf{d}) \ \mid \ (\mathbf{p},\mathbf{d}) \in X \times \mathcal{D} \Big\} \tag{7.29}$$

$$= \left\{ \frac{1}{\sqrt{C}} \sum_{c=0}^{C-1} F_{0,c}(\mathbf{p}) F_{1,c}(\mathbf{p}+\mathbf{d}) \ \mid \ (\mathbf{p},\mathbf{d}) \in X \times \mathcal{D} \right\}, \tag{7.30}$$

where F_0 and $F_1 \in \mathbb{R}^{H \times W \times C}$ are the source and target feature maps, $X = [0,\mathrm{H}(\times [0,\mathrm{W}(\cap \mathbb{N}^2$ refers to the location of a pixel in the source image and $\mathcal{D} = [-R_{\max}, R_{\max}]^2 \cap \mathbb{Z}^2$ corresponds to the displacement range. $R_{\max}$ represents the search radius of the square neighborhood and consequently the maximum displacement along x and y directions. This local correlation volume contains $N(2R+1)^2$ costs, where $N = HW$ is the number of pixels in the images. It may result in an inaccurate optical flow due to the limited search space. Throughout this report, $D = 2R_{\max} + 1$ will refer to the side length of the square search region.

Global correlation volume

RAFT proposed an all-pairs correlation volume that achieved state-of-the-art performance. This full correlation volume is formed by computing the normalized inner product between pairwise feature vectors. Mathematically, it can be formulated as follows:

$$\mathrm{CV} = \Big\{ F_0(\mathbf{p}_0) \cdot F_1(\mathbf{p}_1) \;\mid\; (\mathbf{p}_0, \mathbf{p}_1) \in X^2 \Big\}, \tag{7.31}$$

where F_0 and $F_1 \in \mathbb{R}^{H \times W \times C}$ are the source and target feature maps and $\mathrm{X} = [0,\mathrm{H}(\times [0,\mathrm{W}(\cap \mathbb{N}^2$ refers to the source and target image domain. In order to combine information about both large and small displacements, RAFT constructed a 4-level pyramid by pooling the last two dimensions of the correlation volume. Therefore, for each pyramid level l, the correlation volume CV^l has dimensions $H \times W \times H/2^l \times W/2^l$ where 2^l is the pooled kernel size (see Fig.7.12).

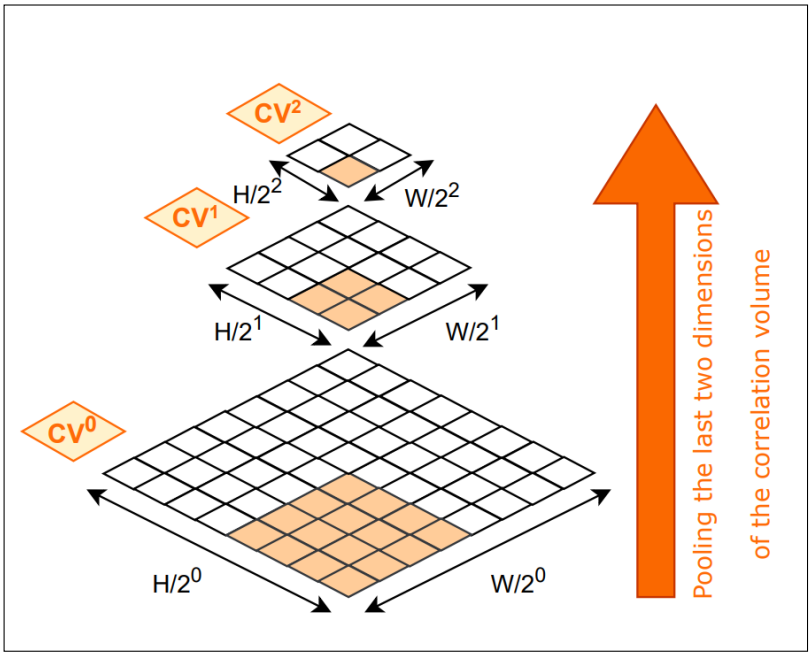
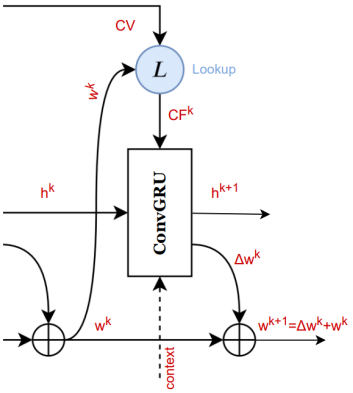


Figure 7.12 – A 3-layer correlation pyramid

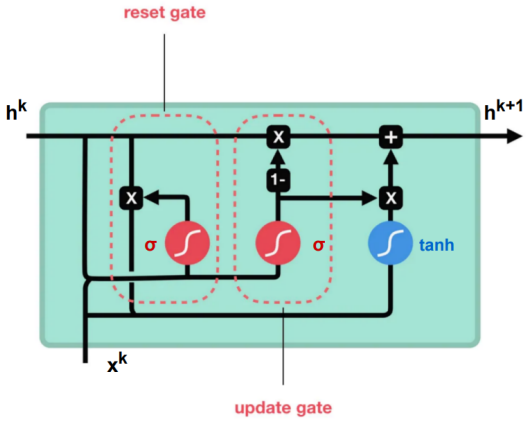
The global correlation volume contains N^2 costs, where $N = HW$ is the number of pixels in the images. For high-resolution images where the height and the width are large, the global approach leads to large memory consumption and high computational cost.

Iterative updates

The recurrent update operator introduced in Fig.7.11 iteratively regresses a residual flow $\Delta\mathbf{w}$ to refine the flow estimate: $\mathbf{w}^{k+1} = \mathbf{w}^k + \Delta\mathbf{w}^k$. It is usually achieved using ConvGRU blocks (Convolutional Gated Recurrent Unit). Its principle is illustrated in Figure 7.13.



(a) GRU-based update operator



(b) Inside a ConvGRU block (source: towardsdatascience.com - Illustrated guide to LSTMs and GRUs)

Figure 7.13 – Principle of the recurrent optical flow update

The flow is generally initialized at $\mathbf{w}^0 = \mathbf{0}$ (except for the DIP method [156] described below, where the flow is randomly initialized). At each iteration k, the current hidden state h^k, the current flow $\mathbf{w}^k$, and the context are injected into the update operator, which further outputs a residual flow $\Delta\mathbf{w}^k$. In RAFT, the correlation feature map CF^k is also fed into the update operator. This map is constructed after every correlation lookup. This operator, which precedes the update operator, is illustrated in Figure 7.14.

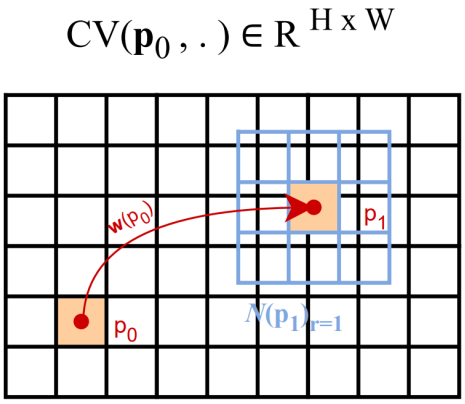


Figure 7.14 – Correlation lookup (r=1)

Given the current estimate of optical flow $\mathbf{w}^k$, each pixel $\mathbf{p}_0$ in I_0 is mapped to its corresponding pixel $\mathbf{p}_1$ in I_1: $\mathbf{p}_1 = \mathbf{p}_0 + \mathbf{w}^k$. A $(2r+1)\times(2r+1)$ square grid $\mathcal{N}_r(\mathbf{p}_1)$ is then

constructed around pixel $\mathbf{p}_1$. Since the coordinates of the estimated optical flow vectors take real values, the square neighborhood of $\mathbf{p}_1$ will also contain real indices. To extract the correlation values of such indices, bilinear sampling is performed on the correlation volumes CV^k. The correlation feature map is then obtained after concatenating the correlation values at different pyramid levels. It has dimensions $(H\ W\ |\mathcal{N}_r|) \times L$, where $|\mathcal{N}_r| = (2r+1)^2$ is the number of neighbors and L is the number of pyramid levels (4 levels in RAFT).

Extensions on the cost volume.

In order to overcome the drawbacks of the global correlation volume involved in RAFT, other methods proposed to improve the processing of cost information. Instead of directly operating on the 4D correlation volume, FlowFormer [143] uses the attention mechanism to encode the entire $H \times W \times H \times W$ correlation volume into a compact $H \times W \times K$ cost memory ($K \ll N$), which better captures information across pixels. Although this method effectively addresses cost information, it still requires to build the entire correlation volume and is therefore heavily memory consuming.

Sparse global correlation volume The method described in [159] introduces a sparse global correlation strategy to reduce the memory consumption of dense correlation computation, while maintaining a global search for correspondences. The Sparse Correlation Volume (SCV) is constructed from k nearest matches in the target feature map for each feature vector in the source feature map. These scores are then stored in a sparse data structure that is further converted into a 2D dense motion tensor via a multi-scale displacement encoder. When building correlation volumes with high-resolution feature maps, this strategy results in significant memory savings in comparison to RAFT (the complexity of correlation computation is reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(Nk)$). However, it fails in blurry and featureless regions where top-k correlations might no be sufficient to include the correct match.

PatchMatch-based correlation volume Instead of using the global sparse strategy that suffers from a loss of accuracy, the DIP method [156] adopts a PatchMatch-based correlation volume. The PatchMatch method was proposed by Barnes et al. [160] to compute patch correspondences in a pair of images. It leverages that neighboring pixels usually have coherent matches. DIP consists of two modules: an inverse propagation

module and a local search module. The first module propagates information from the neighbors once a good match is found. The second module prevents local minima by performing a local search after each propagation. The correlation volume is then defined by:

$$\mathrm{CV} = \Big\{ F_0(\mathbf{p}) \cdot \mathbf{W}(F_2, \mathbf{S}(\mathbf{w}, \Delta\mathbf{s}))(\mathbf{p}) \mid (\mathbf{p}, \Delta\mathbf{s}) \in X \times \mathcal{S} \Big\}, \tag{7.32}$$

where $\mathcal{S} = \Big\{$(1,1), (-1,-1), (0,0), (+1,-1), (-1,+1)$\Big\}$ refers to the shift direction, $\mathbf{S}(\mathbf{w}, \Delta\mathbf{s})$ is the flow shifted to $\Delta\mathbf{s}$, and $\mathbf{W}$ warps the target feature map with the shifted flow. The correlation volume has dimensions $H \times W \times |\mathcal{S}|$, where $|\mathcal{S}|$ is the number of flow candidates (5 in this specific case). DIP accomplished a good trade-off between performance and cost. It also achieved SOTA performances on KITTI-15 dataset and Sintel Clean dataset

The classical propagation stage requires the shift to be run each time the flow is updated. This in turn increases memory operations, especially when working with high-resolution images. That is why DIP proposes an inverse propagation module that shifts the target feature maps in advance instead of shifting the flow after every iteration.

Supervised learning

Let θ be the set of learnable parameters in a given network. These parameters typically include feature extractor and update operator parameters. Let D be the set of image sequences $\{(I_i)_{i=1}^{T}\}$, where T is the number of images in a sequence. We learn the parameters θ by minimizing a loss function $\mathcal{L}$: $\theta^* = \arg\min_{\theta} \mathcal{L}(D, \theta)$.

When ground truth is available, the commonly used error measure is the Average End-Point Error (AEPE) between the predicted flow vector $\mathbf{w}$ and the ground-truth vector $\mathbf{w}_{\mathrm{GT}}$, averaged over all pixels (N pixels):

$$\mathrm{AEPE} = \frac{1}{N} \sum_{\mathbf{p}} \|\mathbf{w}_{\mathrm{GT}}(\mathbf{p}) - \mathbf{w}(\mathbf{p})\|_2, \tag{7.33}$$

where $\|\mathbf{w}_{\mathrm{GT}} - \mathbf{w}\|_2 = \sqrt{\Big(\mathbf{w}_{\mathrm{GT,x}} - \mathbf{w}_x\Big)^2 + \Big(\mathbf{w}_{\mathrm{GT,y}} - \mathbf{w}_y\Big)^2}$.

In multi-scale approaches as in [98], the training loss can be written as follows:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{l=l_0}^{L-1} \alpha_l \sum_{\mathbf{p}} \|\mathbf{w}_{\mathrm{GT}}^l(\mathbf{p}) - \mathbf{w}^l(\mathbf{p})\|_1, \tag{7.34}$$

where $\mathbf{w}^l$ is the flow at the l-th pyramid level, α_l are increasing weights since more

importance is placed on the finest resolution $l = L - 1$. The model starts to predict the flow at the coarsest-resolution l_0 and uses bilinear interpolation to obtain the full-resolution optical flow at level $L - 1$.

The previously described single-resolution approaches [56], [143], [155], [156] output a sequence of flow estimates $\{\mathbf{w}_1, \ldots, \mathbf{w}_M\}$, where M is the number of iterations in the flow refinement. Their training loss can be expressed as:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{k=1}^{M} \alpha_k \sum_{\mathbf{p}} \|\mathbf{w}_{\mathrm{GT},k}(\mathbf{p}) - \mathbf{w}_k(\mathbf{p})\|_1, \tag{7.35}$$

where α_k is computed from Eq.(7.36) with $\gamma = 0.8$ (exponentially increasing weights since more importance is placed on the final optical flow estimate).

$$\alpha_k = \gamma^{N-k} \tag{7.36}$$

Unsupervised learning

When working with real-world sequences, it is difficult to obtain ground-truth labels for dense optical flow. That is why supervised techniques usually rely on synthetic data for training. Therefore, generalization remains challenging due to the potential large gap between the synthetic data they were trained on and the real-world images of a given application. In order to overcome this issue, other works proposed to learn optical flow in an unsupervised way.

Unsupervised methods often combine components from supervised learning regarding the network architecture and classical (variational) optical flow methods regarding the loss function. UnFlow [161] proposes an unsupervised version of FlowNetCorr. ARFlow [99] follows the pipeline of PWC-Net while reducing its number of parameters. This light-weight extension of PWC-Net integrates self-supervision from augmentations into the learning framework in order to generate challenging scenes. These augmentations include spatial, occlusion and appearance transformations. SMURF [162] extends the RAFT architecture to an unsupervised setting by performing self-supervision in a sequence-aware manner. It handles out-of-frame motion and occlusions thanks to novel ideas such as full-image warping and multi-frame flow refinement. A recent method [163] improves supervised flow estimation by coupling classical supervised training with an unsupervised criterion based on the brightness constancy equation. Their method allows to do semi-supervised training where only a part of the data is annotated, which is an asset for

unsupervised domain adaptation.

The key difference between classical optical flow methods and unsupervised methods is that the former performs an optimization per image pairs, while the latter optimizes the loss function across the whole training set. This type of optimization allows information to be shared between image pairs, which can improve the performance of a model.

Data term Like many classical methods [10], unsupervised methods rely on the brightness constancy assumption (BCA), which states that the pixel intensity does not change as it moves. Thus, the primary component of the loss function is photometric consistency. Its goal is to minimize the difference between the source image I_0 and the flow-warped target image I_1. It can be expressed as follows:

$$\mathcal{L}_{sim}(D, \theta) = \frac{1}{N} \sum_{\mathbf{p}} \phi\Big(f_D(I_0(\mathbf{p}), I_1(\mathbf{p}'))\Big), \tag{7.37}$$

where ϕ is a penalty function (e.g., the L2 norm, the L1 norm, the Huber function), f_D is the displaced frame difference ($f_D(I_0(\mathbf{p}), I_1(\mathbf{p}')) = I_1(\mathbf{p}') - I_0(\mathbf{p})$), and $\mathbf{p}' = \mathbf{p} + \mathbf{w}(\mathbf{p})$.

The BCA may be violated in several configurations. As a result, some works have proposed to replace the f_D measure with more reliable ones such as the Census loss, which is invariant to additive and multiplicative illumination changes. The Structural Similarity Measure (SSIM) could also be used.

The similarity loss is violated when pixels are occluded or moved out of view between two consecutive frames. To enforce photometric consitency only for non-occluded pixels we have to design an occlusion-aware photometric loss:

$$\mathcal{L}_{sim}(D, \theta) = \frac{1}{N} \sum_{\mathbf{p}} O(\mathbf{p}) \cdot \phi\Big(f_D(I_0(\mathbf{p}), I_1(\mathbf{p}'))\Big), \tag{7.38}$$

where O is a binary occlusion mask that disables photometric consistency for occluded pixels.

Regularization term In unsupervised learning, the loss function must involve a form of regularization. Indeed, photometric loss is ambiguous in textureless regions. Similarly to classical methods, it is possible to incorporate an edge-aware first order smoothness

term into the overall unsupervised loss function:

$$\mathcal{L}_{smoo} = \frac{1}{N} \sum_{u \in \{x,y\}} \sum_{\mathbf{p}} \left\| \nabla_u \mathbf{w}(\mathbf{p}) \right\|_1 . e^{-\alpha_{smoo} |\nabla_{\mathbf{u}} I_0(\mathbf{p})|}, \tag{7.39}$$

where $\left\| \nabla_u \mathbf{w}(\mathbf{p}) \right\|_1 = \left| \frac{\partial \mathbf{w}_x}{\partial x}(\mathbf{p}) \right| + \left| \frac{\partial \mathbf{w}_x}{\partial y}(\mathbf{p}) \right| + \left| \frac{\partial \mathbf{w}_y}{\partial x}(\mathbf{p}) \right| + \left| \frac{\partial \mathbf{w}_y}{\partial y}(\mathbf{p}) \right|$.

Intensity edge sensitivity in Eq.(7.39) is weighted by $\alpha_{smoo} > 0$:

- if $\alpha_{smoo} \to 0$, smoothness loss does not take into account image edges,
- if $\alpha_{smoo} \to +\infty$, the smoothness term tends to zero and is therefore neglected in the regularization term,
- hence, α_{smoo} must be high enough to acknowledge edge sensitivity, while not exceeding a certain threshold so that it allows for smoothness regularization. This threshold depends on the data we are dealing with.

Limitations of state-of-the-art deep OF methods

In subsection 7.4.2, we presented RAFT, a supervised deep network architecture for optical flow that achieved state-of-the art performance on major datasets such as KITTI and MPI Sintel, when it was first released. Although its dense global correlation volume is effective for accurate optical flow estimation, it leads to large memory consumption and heavy computation, especially for high-resolution large-scale images. Supervised methods like DIP [156] and SCV [159] have proposed alternatives to the global correlation volume, focusing on finding a good compromise between accuracy on one hand and low memory consumption and computational cost on the other hand. However, these methods still fail to capture the motion of fine structures and small moving objects in satellite images, which we will show in Sec.7.4.5. We believe that this is due to the fact that all of these methods use low-resolution feature maps. In RAFT, the all-pair correlation volume comes at a cost. It stores the matching costs within the entire displacement range, and therefore, requires the feature maps to be downsampled by $\frac{1}{8}$. DIP and SCV reduce the resolution of feature maps by $\frac{1}{4}$. In addition, we need to develop an unsupervised OF method, since no OF ground truth is available for the satellite images.

7.4.3 SoFlow: Unsupervised optical flow method for small moving objects

We propose SoFlow, an unsupervised method for estimating the optical flow of small moving objects in high-resolution images. First, we will describe the network architecture and the loss function that we have defined. Then, we will show how our contributions enable the deployment of the model for satellite imagery.

SoFlow overall framework

Network architecture As mentioned in subsection 7.4.2, unsupervised methods usually start from networks proposed by supervised methods. Since RAFT inspired several deep learning networks for optical flow estimation such as DIP or SMURF, we found it relevant to customize RAFT architecture for an effective motion estimation of small moving objects. Then, our network consists of three main components:

1. *Feature and context encoder*: We replace the $\frac{1}{8}$ downsampling induced by the CNN-based encoder with a $\frac{1}{2}$ downsampling by modifying the stride parameter in RAFT original encoder. This choice is motivated by the size of the observed vehicles in the satellite images.

2. *Correlation block*: Starting from the observation that the displacement range of moving objects is constrained in satellite images, we replace the global correlation volume with a local one. The former computes an all-pair visual similarity, while the latter computes the visual similarity between each pixel $\mathbf{p}$ in I_0 within a radius R in I_1. By restricting the search space for optical flow, we reduce the memory consumption and computational cost of correlation computation from $\mathcal{O}(N^2)$ to $\mathcal{O}(N(2R+1)^2)$. We also construct a correlation pyramid by pooling the last two dimensions of the correlation volume. The first level of this pyramid captures the smallest displacements, while the highest level gives information about larger displacements.

3. *Iterative update*: Similarly to RAFT, this stage includes three main tasks, namely correlation lookup (L), GRU-based residual flow estimation, and optical flow upsampling (U). At each iteration, the half-size predicted flow is upsampled to match the resolution of the ground truth.

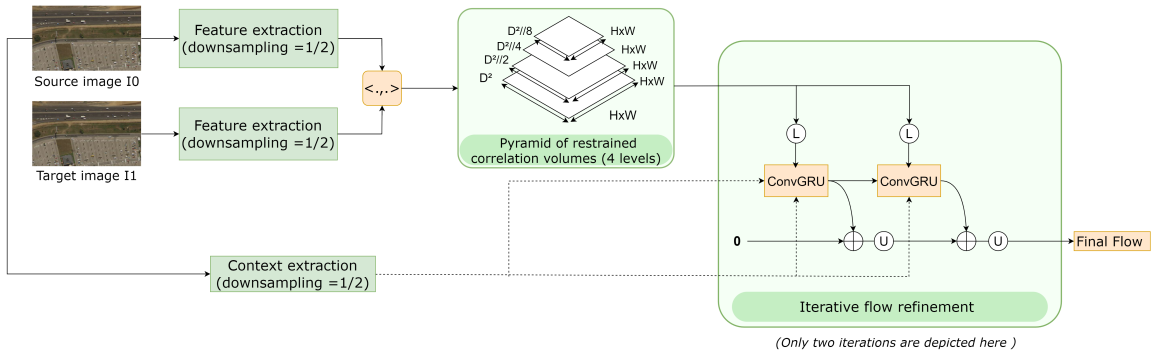


Figure 7.15 – SoFlow architecture

Loss function Our unsupervised loss function is defined as follows:

$$\mathcal{L}(D,\theta) = \underbrace{\omega_{sim}\ \mathcal{L}_{sim}(D,\theta)}_{\text{Data term}} + \underbrace{\omega_{smoo}\ \mathcal{L}_{smoo}(D,\theta) + \omega_{spar}\mathcal{L}_{spar}(D,\theta)}_{\text{Regularization terms}}, \tag{7.40}$$

which is a weighted combination of three terms, a similarity loss term $\mathcal{L}_{\text{sim}}$, an edge-aware smoothness term $\mathcal{L}_{\text{smoo}}$, and a sparsity term $\mathcal{L}_{\text{spar}}$. As a matter of fact, the sparsity term was a late addition to the loss function. Therefore, the experimental results reported below were obrained without the sparsity term.

The similarity and smoothness loss terms are similar to the ones we presented in Eq.(7.38) and Eq.(7.39) respectively, except we discard occlusion reasoning. The third term is formulated as follows:

$$\mathcal{L}_{\text{spar}} = \frac{1}{N}\sum_{\mathbf{p}} \|\mathbf{w}(\mathbf{p})\|_2 . e^{-\alpha_{\text{spar}}|\frac{\partial I}{\partial t}(\mathbf{p})|}, \tag{7.41}$$

where

$$|\frac{\partial I}{\partial t}(\mathbf{p})| \approx \frac{1}{|\mathcal{V}(\mathbf{p})|}\sum_{j\in\mathcal{V}(i)} \|I_1(\mathbf{p},t) - I_0(\mathbf{p},t)\|_1 \tag{7.42}$$

where $\mathcal{V}(\mathbf{p})$ is a local neighborhood of $\mathbf{p}$.

Local correlation volume with manual backward implementation

In order to reduce the memory consumption induced by the correlation block and to improve its performance, we propose an efficient way to compute the local cost volume and the gradient of the loss function with respect to the feature maps. Let us recall the

mathematical expression of the local correlation volume (CV):

$$\mathrm{CV} = \Big\{ F_0(\mathbf{p}) \cdot F_1(\mathbf{p}+\mathbf{d}) \;\mid\; (\mathbf{p},\mathbf{d}) \in X \times \mathcal{D} \Big\} \tag{7.43}$$

$$= \left\{ \frac{1}{\sqrt{C}} \sum_{c=0}^{C-1} F_{0,c}(\mathbf{p}) F_{1,c}(\mathbf{p}+\mathbf{d}) \;\mid\; (\mathbf{p},\mathbf{d}) \in X \times \mathcal{D} \right\}, \tag{7.44}$$

Efficient computation of the correlation volume To effectively build the cost volume, we adopt a per-shift computation rather than a per-pixel computation:

- *Per-pixel computation.* For each pixel $\mathbf{p}$ in I_0, one extract and store the values of its neighbors in I_1 before computing their dot product. This approach considers the 4D local correlation volume $\mathrm{CV} \in \mathbb{R}^{D^2 \times H \times W}$ as a stack of $H \times W$ correlation maps of dimension D^2.
- *Per-shift computation.* For each neighboring position $\mathbf{d}$ in $\mathcal{D} = [-R, R]^2 \cap \mathbb{Z}^2$, we compute the correlation between pixels $\mathbf{p}$ in I_0 and their neighbors at position $\mathbf{p}+\mathbf{d}$ in I_1. These neighbors are obtained by shifting the target feature map by $\mathbf{d}$. This approach allows us to formulate the 4D correlation volume CV as a stack of D^2 correlation maps $\mathrm{CV}_{d'}$ of dimensions $H \times W$, each representing the correlation between the source feature map and the d-shifted target feature map. The integer d' refers to a pointer that scans the grid $\mathcal{D}$ in lexicographic order.
  In the following, each shift vector $\mathbf{d}$ will be mapped by g_{map} to its corresponding index given by the integer value d':

$$g_{\text{map}} : \quad \mathcal{D} \quad \longrightarrow \{0, \ldots, D^2 - 1\} \tag{7.45}$$

$$\mathbf{d} = (d_x, d_y) \;\longmapsto\; d' \tag{7.46}$$

  This mapping is designed such that the first cost volume CV_0 stores the similarity costs between pixels $\mathbf{p}$ in I_0 and their top-left neighbor $\mathbf{p}+\mathbf{d} = (x - R, y - R)$ in I_1, while the last cost volume CV_{D^2-1} stores the correlations between pixels $\mathbf{p}$ in I_0 and their bottom-right neighbor $\mathbf{p}+\mathbf{d} = (x + R, y + R)$ in I_1. Also, when computing the similarity between a pixel $\mathbf{p}$ in I_0 and its corresponding pixel at the same position in I_1, there is no need to shift the target feature map, which implies that $d' = \frac{D^2-1}{2}$ and $\mathbf{d} = (0, 0)$.

```python
import torch
from shapecheck.ShapeCheck import ShapeCheck

def _compute_local_corr_volume(fmap0: torch.Tensor, fmap1: torch.Tensor, radius: int):
    b, c, h, w = fmap0.shape
    D = 2 * radius + 1
    norm = torch.sqrt(torch.tensor(c))
    sc = ShapeCheck(fmap0.shape, 'b c h w')
    sc.update([D, D],'pH pW')
    unfold = torch.nn.Unfold(kernel_size=D, padding=(radius,radius))
    unfolded_fmap1 = sc.rearrange(unfold(fmap1), 'b (c pH pW) (h w) -> b c (pH pW) (h w)')
    flat_fmap0 = sc.rearrange(fmap0, 'b c h w -> b c (h w)')
    corr = torch.einsum('Bcm,Bcpm', flat_fmap0, unfolded_fmap1)/norm
    corr = sc.rearrange(corr, 'b (pH pW) (h w) -> b h w pH pW')
    return corr
```

Listing 1 – Per-pixel computation

Listing 1 introduces the straightforward approach of computing the local correlation volume. For each pixel **p** in I_0, we extract the neighbourhood of its corresponding pixel **p'** in I_1 and compute the correlation between **p** and all the pixels within the local neighbourhood of **p'**. This computation involves the `torch.nn.Unfold` class, an operation that extracts sliding local blocks from the batched input tensor `fmap1`. For an input tensor of shape $B \times C \times H \times W$ and a $D \times D$ kernel, the operation outputs a tensor of shape $B \times (CD^2) \times L$, where D^2 is the total number of values within each block and L is the total number of such blocks (HW in our case). The main drawback of the `Unfold` operation is that it extracts the values in the local blocks by **copying** them from the large input tensor. This considerably increases the memory usage, and makes this approach unusable on high-resolution images. An alternative to the *per_pixel computation* is the *per_shift computation* presented in listing 2.

```python
import torch
from shapecheck.ShapeCheck import ShapeCheck
import torchvision.transforms as T

def _compute_lazy_local_corr_volume(fmap0: torch.Tensor, fmap1: torch.Tensor, radius: int):
    b, c, h, w = fmap0.shape
    D = 2 * radius + 1
    norm = torch.sqrt(torch.tensor(c))
    sc = ShapeCheck(fmap1.shape, 'b c h w')
    sc.update([D, D],'pH pW')
    padded_fmap1 = T.Pad(padding=radius)(fmap1)
    corr = torch.zeros(b,D**2,h,w, device = fmap0.device)
    for d in range(0,D**2) : # loop over shift
        dx, dy = d//D,d%D
        corr[:,d,...] = ((fmap0 * padded_fmap1[:,:,dx:h+dx,dy:w+dy]).sum(axis=1))/norm
    corr = sc.rearrange(corr, 'b (pH pW) h w -> b h w pH pW')
    return corr
```

Listing 2 – Per-shift computation

A per-shift computation is less memory consuming, since it directly operates on the whole feature maps, and does not require to store the features of the neighbors for every pixel in the source image. Also, instead of building the tensor `unfolded_fmap1` of shape $B \times C \times (2R+1)^2 \times (HW)$, we construct a smaller tensor `padded_fmap1` of shape $B \times C \times (H+2R) \times (W+2R)$. Following this approach, we need to pad the target feature on all sides with a "pad" value of R, and compute the matching costs for each shifting position.

Manual backward implementation The graph structure of our local correlation block can slow down the computations performed by the autograd engine (see Fig.7.16). Preliminary experiments on the correlation volume also showed that its backward is very slow. In order to improve performance during training, we implement our own customized backpropagation function. This manual backward requires us to express the derivatives of the local correlation volume with respect to its input, namely the source and target feature maps. Therefore, given the output gradient $\nabla_{\mathbf{CV}}\mathcal{L}$, we want to compute the input gradients $\nabla_{F_0}\mathcal{L}$ and $\nabla_{F_1}\mathcal{L}$.

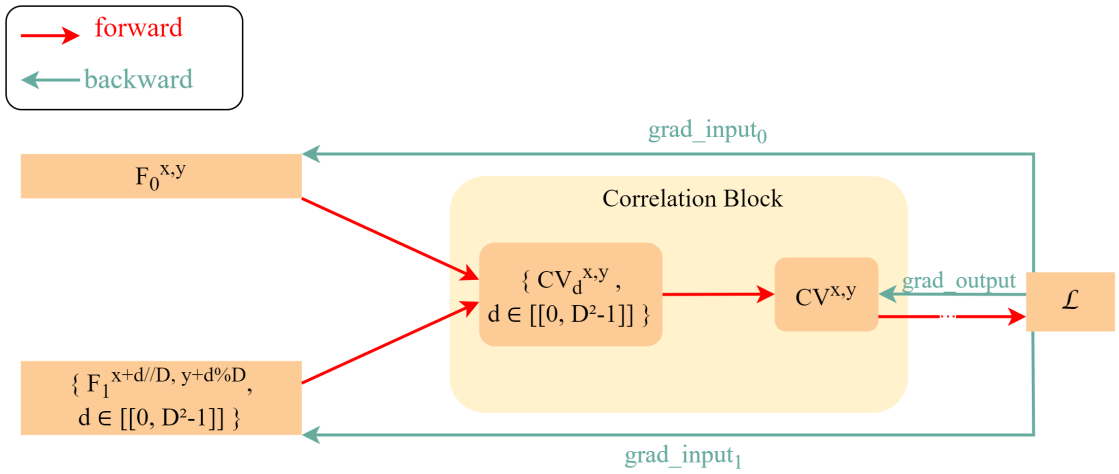


Figure 7.16 – Computation graph of the correlation block

Figure 7.16 illustrates the backward pass for a single pixel $\mathbf{p} = (x, y)$ in the source image I_0. In the backward pass, we receive a tensor containing the gradient of the loss with respect to the output ($grad\_output$), and we need to compute the gradient of the loss with respect to the input ($grad\_input = (grad\_input_0, grad\_input_1)$). Let us note that both $grad\_input_k$ have dimensions $C \times H \times W$, and $grad\_output$ has dimensions $D^2 \times H \times W$.

Let us first fix a pixel location (x, y) in the source image and compute the gradient of the loss *w.r.t.* F_0 and F_1. Following Fig.7.16, we have:

$$grad\_input_k^{x,y} = \frac{\partial \mathcal{L}}{\partial F_k^{x,y}} = \sum_{d=0}^{D^2-1} \sum_{i,j} \frac{\partial \mathcal{L}}{\partial C_d^{i,j}} \frac{\partial \mathrm{CV}_d^{i,j}}{\partial F_k^{x,y}}. \tag{7.47}$$

Replacing (i, j) and (x, y) by $\mathbf{q}$ and $\mathbf{p}$ to keep the notation uncluttered, we have:

$$grad\_input_k^{\mathbf{p}} = \frac{\partial \mathcal{L}}{\partial F_k^{\mathbf{p}}} = \sum_{d=0}^{D^2-1} \sum_{\mathbf{q}} \frac{\partial \mathcal{L}}{\partial C_d^{\mathbf{q}}} \frac{\partial \mathrm{CV}_d^{\mathbf{q}}}{\partial F_k^{\mathbf{p}}}. \tag{7.48}$$

The correlation volume can also be expressed as follows:

$$\mathrm{CV}_d^{\mathbf{q}} = F_0(\mathbf{q}) \cdot F_1(\mathbf{q} + \mathbf{shift}_d), \tag{7.49}$$

where $\mathbf{shift}_d = g_{\text{map}}^{-1}(d)$ is the shifting coordinates that correspond to integer d.

- *Gradient of the loss w.r.t the source feature map* F_0. For a given shift d, we have:

$$\frac{\partial C_d^{\mathbf{q}}}{\partial F_0^{\mathbf{p}}} = \begin{cases} F_1(\mathbf{q} + \mathbf{shift}_d) & \text{if } \mathbf{q} = \mathbf{p} \\ 0 & \text{otherwise.} \end{cases} \tag{7.50}$$

Back to Eq.(7.48), we can conclude that:

$$grad\_input_0^{\mathbf{p}} = \frac{\partial \mathcal{L}}{\partial F_0^{\mathbf{p}}} = \sum_{d=0}^{D^2-1} \underbrace{\frac{\partial \mathcal{L}}{\partial C_d^{\mathbf{p}}}}_{grad\_ouput[d,\mathbf{p}]} F_1^{\mathbf{p}+\mathbf{shift}_d}. \tag{7.51}$$

- *Gradient of the loss w.r.t the target feature map* F_1. For a given shift d:

$$\frac{\partial C_d^{\mathbf{q}}}{\partial F_1^{\mathbf{p}}} = \begin{cases} F_0(\mathbf{q}) & \text{if } \mathbf{q} = \mathbf{p} - \mathbf{shift}_d \\ 0 & \text{otherwise.} \end{cases} \tag{7.52}$$

Back to Eq.(7.48), we can conclude that:

$$grad\_input_1^{\mathbf{p}} = \frac{\partial \mathcal{L}}{\partial F_1^{\mathbf{p}}} = \sum_{d=0}^{D^2-1} \underbrace{\frac{\partial \mathcal{L}}{\partial C_d^{\mathbf{p}-\mathbf{shift}_d}}}_{grad\_ouput[d,\mathbf{p}-\mathbf{shift}_d]} F_0^{\mathbf{p}-\mathbf{shift}_d}. \tag{7.53}$$

Like the architecture itself, the backward is implemented in Pytorch. The implementation details are shown in Fig.7.16.

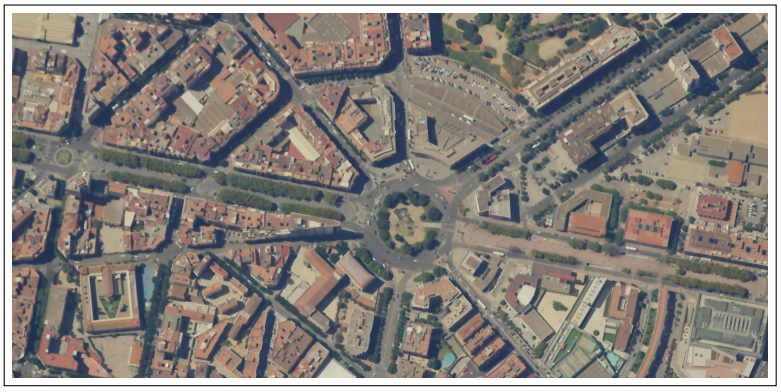
7.4.4 Technical details

We train our model using the loss function defined in subsection 7.4.3. For the similarity loss term, we use SSIM, and the hyperparameters are set as follows: $\omega_{\text{sim}} = 1$, $\omega_{\text{smoo}} = 1$ and $\alpha_{\text{smoo}} = 20$. So far, we have not investigated the influence of sparsity on the performance of our model, which we keep for future work. The network is trained on a subset of 40 satellite sequences. We keep other sequences for the test and for qualitative (visual) evaluation in order to assess the model generalization ability.

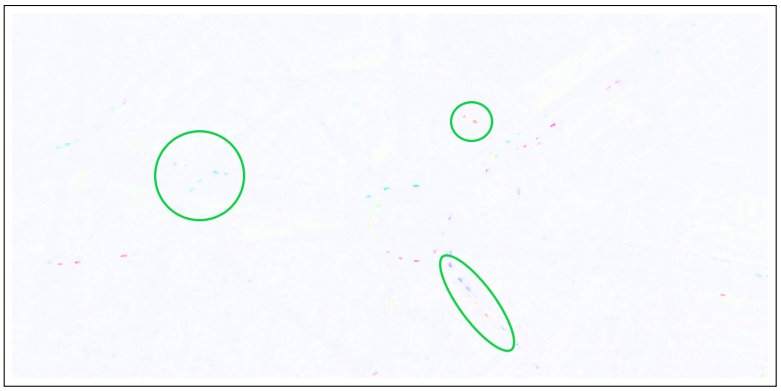
7.4.5 Experimental results

We report sample results of our SoFlow method (without the sparsity term) on satellite images, and compare them with two supervised deep learning methods, DIP [156] and SCV [159]. As a matter of fact, these images, provided by Airbus Defence&Space, are aerial images acquired in such a way that they simulate the resolution of the (future) satellite images. In addition, they are stabilized.

Figures 7.17 and 7.18 depict a visual comparison of optical flow estimates on the test sequence PlacaTarraco and the training sequence A6C15-Glories, respectively. In order to make a consistent comparison, we use an absolute normalization of the flow before outputting its HSV color representation. For each sequence, we normalized the flow by an approximation of its maximum flow magnitude.

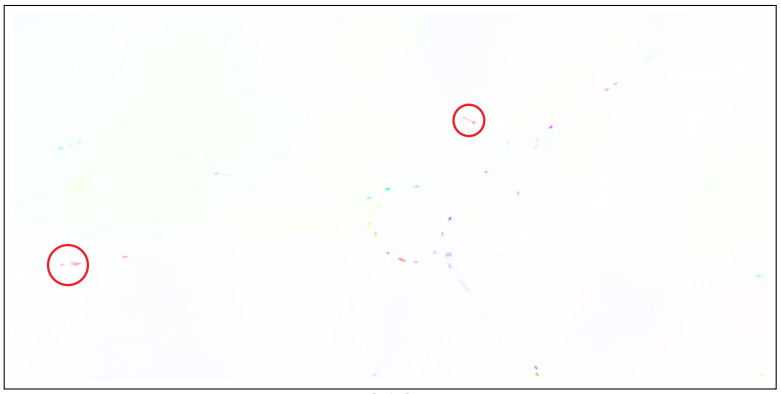


(a) Input image



(b) Ours - SoFlow

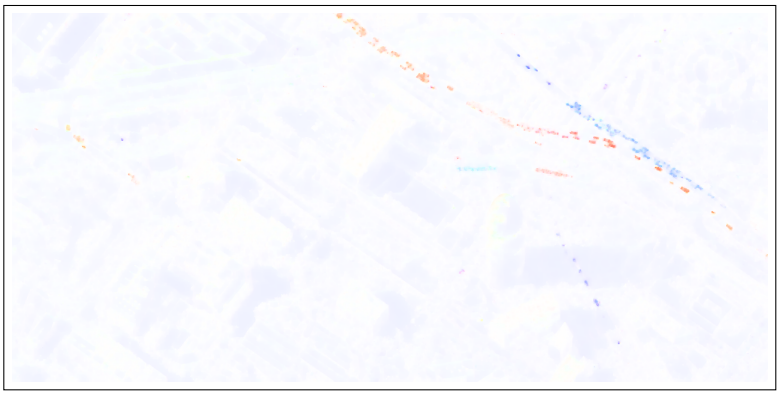
(c) SCV



(d) DIP

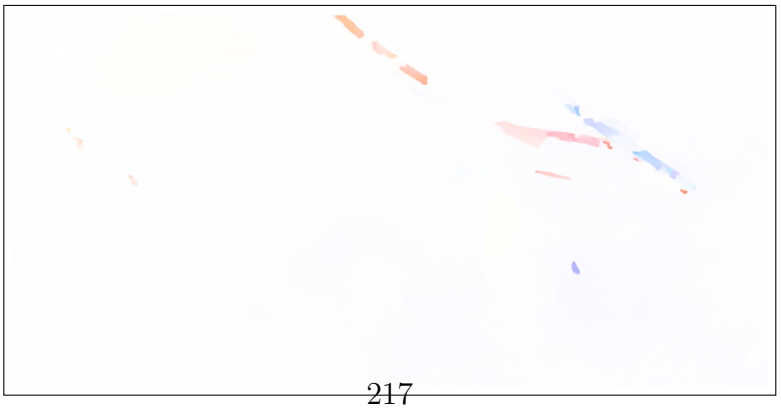
Figure 7.17 – Optical flow computed on PlacaTarraco (between frames 8 and 9) for the three methods. From top to bottom, one image of the sequence, our method SoFlow, SCV method [159] and DIP method [156].

(a) Input image



(b) Ours

(c) SCV



(d) DIP

Figure 7.18 – Optical flow computed on A6C15-Glories (between frames 7 and 8) for the three methods. From top to bottom, one image of the sequence, our method SoFlow, SCV method [159] and DIP method [156].

The visual comparison between the OF results of the three methods, SoFlow, SCV and DIP, reported in Fig.7.17 and 7.18 confirms that our method outperforms the supervised SCV and DIP methods, while being unsupervised. It is able to distinctly capture the motion of the small moving objects in the high-resolution satellite images. In return, results are a bit noisy in the background. In Fig.7.17, SCV omits most of the small moving vehicles, while DIP achieves better performance than SCV for some fine structure areas. However, DIP blends the optical flow of adjacent vehicles as shown in the red circled regions in Figure 7.17(d). This blurred OF estimation for DIP and SCV is even more obvious in Fig.7.18.

As for RAFT, we believe that this tendency is due to the $\frac{1}{4}$ downsampling in the feature extraction module. By building higher-resolution feature maps, our method alleviates this problem, since we can clearly distinguish close moving vehicles, as highlighted in the green circled areas in Fig.7.17(b), or even more clearly in Fig.7.18. Nevertheless, one weakness of our approach is the handling of the (stabilized) background region. Indeed, inference on both PlacaTarraco and A6C15-Glories sequences shows that the model often predicts a non-zero motion in the stabilized background. One assumption is that our loss function does not handle well enough featureless regions. Indeed, the features of such regions may have a large number of apparently correct matches due to the inherent ambiguity, and it is therefore difficult to quantify how much the flow-warped target image is deviated from the source image. The sparsity term should allow us to correct this, by penalizing non-zero flows in the stabilized background.

7.4.6 Partial Conclusion

We have presented several contributions to improve the computation of optical flow in high-resolution satellite imagery. We have designed the SoFlow method, an unsupervised model that integrates a novel local correlation volume with efficient computational cost and reduced memory consumption. SoFlow is able to correctly handle small moving objects in large-scale satellite image sequences, and outperforms supervised optical flow methods. This confirms that local correlation blocks are suitable for the computation of the optical flow field on satellite images. Furthermore, the fact that we obtain better results with unsupervised learning on real images than with supervised learning on synthetic images suggests that training the model on satellite images helps to obtain more accurate flow fields, as it could be expected.

In future work, we will include the sparsity term in the loss function as described in

subsection 7.4.3. A more advanced approach would be to learn an optical flow prior, as done in [164], which would help to regularize the flow prediction depending on the satellite images seen at training time. Another interesting perspective would be to introduce an attention mechanism that makes the network focus on meaningful areas (e.g., cars in satellite images), and allows to reduce the computational load on uniform areas that are unlikely to move, while maintaining good accuracy on small moving objects.

a

Publication List

1. E. Meunier and P. Bouthemy. Unsupervised motion segmentation in one go: Smooth long-term model over a video (Oct. 2023, https://arxiv.org/abs/2310.01040)
2. E. Meunier and P. Bouthemy. Unsupervised space-time network for temporally-consistent segmentation of multiple motions, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR'2023), Vancouver, June 2023 (highlight), DOI: 10.1109/CVPR52729.2023.02120.
3. E. Meunier, A. Badoual and P. Bouthemy. EM-driven unsupervised learning for efficient motion segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(4):4462-4473, April 2023, DOI: 10.1109/TPAMI.2022.3198480.
4. E. Meunier and P. Bouthemy. Unsupervised computation of salient motion maps from the interpretation of a frame-based classification network, British Machine Vision Conference (BMVC'2021), November 2021, HAL-03469574 .
5. E. Meunier, EM-driven network for efficient unsupervised motion segmentation, Workshop on Mathematics and Image Analysis (MIA'2023), Berlin, February 2023 (abstract).
6. E. Meunier and P. Bouthemy. Apprentissage profond non supervisé fondé sur l'algorithme EM pour la segmentation du mouvement. Conf. AFRIF Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP'2022), Vannes, July 2022.
7. E. Meunier and P. Bouthemy. Localisation de mouvements saillants dans des cartes de flot optique par l'interprétation d'un réseau de classification. Journées des jeunes chercheurs en vision par ordinateur (ORASIS'2021), HAL-INRIA-03339650, Saint Ferréol, September 2021.

BIBLIOGRAPHY

[1] P. Bideau, « Motion segmentation - segmentation of independently moving objects in video », Ph.D. dissertation, University of Massachusetts Amherst, 2020.

[2] S. Williams, Z. Zhao, A. Hafeez, *et al.*, « The discerning eye of computer vision: Can it measure Parkinson's finger tap bradykinesia? », *Journal of the Neurological Sciences*, vol. 416, p. 117 003, 2020.

[3] S. Choudhury, L. Karazija, I. Laina, A. Vedaldi, and C. Rupprecht, « Guess what moves: Unsupervised video and image segmentation by anticipating motion », *in 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*, BMVA Press, 2022, p. 554.

[4] A. Mahendran, J. Thewlis, and A. Vedaldi, « Self-supervised segmentation by grouping optical-flow », *in Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, L. Leal-Taixé and S. Roth, Eds., ser. Lecture Notes in Computer Science, vol. 11133, Springer, 2018, pp. 528–534.

[5] R. Liu, Z. Wu, S. X. Yu, and S. Lin, « The emergence of objectness: Learning zero-shot segmentation from videos », *in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 13 137–13 152.

[6] M. Caron, H. Touvron, I. Misra, *et al.*, « Emerging properties in self-supervised vision transformers », *in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, IEEE, 2021, pp. 9630–9640.

[7] S. Hershey, S. Chaudhuri, D. P. W. Ellis, *et al.*, « CNN architectures for large-scale audio classification », *in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, IEEE, 2017, pp. 131–135.

[8] J. Pennington, R. Socher, and C. Manning, « GloVe: Global vectors for word representation », *in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543.

[9] E. Vahdani and Y. Tian, « Deep learning-based action detection in untrimmed videos: A survey », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, *4*, pp. 4302–4320, 2023.

[10] B. K. Horn and B. G. Schunck, « Determining optical flow », *Artificial intelligence*, vol. 17, *1-3*, pp. 185–203, 1981.

[11] K. Zhang and Z. Chen, « Video saliency prediction based on spatial-temporal two-stream network », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, *12*, pp. 3544–3557, 2018.

[12] W. Kim and C. Kim, « Spatiotemporal saliency detection using textural contrast and its applications », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, *4*, pp. 646–659, 2013.

[13] B. Cheng, A. G. Schwing, and A. Kirillov, « Per-pixel classification is not all you need for semantic segmentation », *in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 17 864–17 875.

[14] T. Zhou, F. Porikli, D. J. Crandall, L. V. Gool, and W. Wang, « A survey on deep learning technique for video segmentation », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, *6*, pp. 7099–7122, 2023.

[15] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. H. Gross, and A. Sorkine-Hornung, « A benchmark dataset and evaluation methodology for video object segmentation », *in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 724–732.

[16] P. Tokmakov, K. Alahari, and C. Schmid, « Learning motion patterns in videos », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 531–539.

[17] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, « SegFlow: Joint learning for video object segmentation and optical flow », *in IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, pp. 686–695.

[18] S. D. Jain, B. Xiong, and K. Grauman, « FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 2117–2126.

[19] T. Zhou, J. Li, S. Wang, R. Tao, and J. Shen, « MATNet: Motion-Attentive Transition Network for Zero-Shot Video Object Segmentation », *IEEE Transactions on Image Processing*, vol. 29, pp. 8326–8338, 2020.

[20] A. Dave, P. Tokmakov, and D. Ramanan, « Towards segmenting anything that moves », *in 2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, IEEE, 2019, pp. 1493–1502.

[21] L. Lian, Z. Wu, and S. X. Yu, « Bootstrapping objectness from videos by relaxed common fate and visual grouping », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, IEEE, 2023, pp. 14 582–14 591.

[22] H. Lamdouar, C. Yang, W. Xie, and A. Zisserman, « Betrayed by motion: Camouflaged object discovery via motion segmentation », *in Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers, Part II*, H. Ishikawa, C.-L. Liu, T. Pajdla, and J. Shi, Eds., ser. Lecture Notes in Computer Science, vol. 12623, Springer, 2020, pp. 488–503.

[23] H. Song, W. Wang, S. Zhao, J. Shen, and K.-M. Lam, « Pyramid dilated deeper ConvLSTM for video salient object detection », *in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science, vol. 11215, Springer, 2018, pp. 744–760.

[24] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli, « See more, know more: Unsupervised video object segmentation with co-attention siamese networks », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 3623–3632.

[25] H. Lamdouar, W. Xie, and A. Zisserman, « Segmenting invisible moving objects », *in 32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, BMVA Press, 2021, p. 231.

[26] J. Xie, W. Xie, and A. Zisserman, « Segmenting moving objects via an object-centric layered representation », *in NeurIPS*, 2022.

[27] Y. J. Koh and C.-S. Kim, « Primary object segmentation in videos based on region augmentation and reduction », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 7417–7425.

[28] S. Ayer and H. Sawhney, « Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding », *in Proceedings of IEEE International Conference on Computer Vision*, 1995, pp. 777–784.

[29] J. Wang and E. Adelson, « Representing moving images with layers », *IEEE Transactions on Image Processing*, vol. 3, *5*, pp. 625–638, Sept./1994.

[30] P. Bouthemy and E. Francois, « Motion segmentation and qualitative dynamic scene analysis from an image sequence », *International Journal of Computer Vision*, vol. 10, *2*, pp. 157–182, 1993.

[31] J.-M. Odobez and P. Bouthemy, « MRF-Based motion segmentation exploiting a 2D motion model robust estimation », *in Proceedings., International Conference on Image Processing*, vol. 3, Washington, DC, USA: IEEE Comput. Soc. Press, 1995, pp. 628–631.

[32] Y. Weiss and E. Adelson, « A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models », *in Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1996, pp. 321–326.

[33] J.-M. Perez-Rua, T. Crivelli, P. Perez, and P. Bouthemy, « Hierarchical motion decomposition for dynamic scene parsing », *in 2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA: IEEE, 2016, pp. 3952–3956.

[34] D. Cremers and S. Soatto, « Motion competition: A variational approach to piecewise parametric motion segmentation », *International Journal of Computer Vision*, vol. 62, pp. 249–265, 2005.

[35] C. Vázquez, A. Mitiche, and R. Laganière, « Joint multiregion segmentation and parametric estimation of image motion by basis function representation and level set evolution », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, *5*, pp. 782–793, 2006.

[36] L. Karazija, S. Choudhury, I. Laina, C. Rupprecht, and A. Vedaldi, « Unsupervised multi-object segmentation by predicting probable motion patterns », *in NeurIPS*, 2022.

[37] S. Shrestha, M. A. Armin, H. Li, and N. Barnes, *Learning To Segment Dominant Object Motion From Watching Videos*, 2021.

[38] M. Narayana, A. R. Hanson, and E. G. Learned-Miller, « Coherent motion segmentation in moving camera videos using optical flow orientations », *in IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, IEEE Computer Society, 2013, pp. 1577–1584.

[39] P. Bideau and E. G. Learned-Miller, « It's moving! A probabilistic model for causal motion segmentation in moving camera videos », *in Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, the Netherlands, October 11-14, 2016, Proceedings, Part VIII*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., ser. Lecture Notes in Computer Science, vol. 9912, Springer, 2016, pp. 433–449.

[40] P. Bideau, A. RoyChowdhury, R. R. Menon, and E. G. Learned-Miller, « The best of both worlds: Combining CNNs and geometric constraints for hierarchical motion segmentation », *in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 508–517.

[41] P. Bideau, R. R. Menon, and E. G. Learned-Miller, « MoA-Net: Self-supervised motion segmentation », *in Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part VI*, L. Leal-Taixé and S. Roth,

Eds., ser. Lecture Notes in Computer Science, vol. 11134, Springer, 2018, pp. 715–730.

[42] P. Bideau, E. Learned-Miller, C. Schmid, and K. Alahari, « The Right Spin: Learning Object Motion from Rotation-Compensated Flow Fields », *ArXiv preprint*, vol. abs/2203.00115, 2022.

[43] A. Ranjan, V. Jampani, L. Balles, *et al.*, « Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 12 240–12 249.

[44] M. Irani and P. Anandan, « A unified approach to moving object detection in 2D and 3D scenes », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, *6*, pp. 577–589, 1998.

[45] Y. Weiss, « Smoothness in layers: Motion segmentation using nonparametric mixture estimation », *in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 520–526.

[46] C. Yang, H. Lamdouar, E. Lu, A. Zisserman, and W. Xie, « Self-supervised video object segmentation by motion grouping », *in 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, IEEE, 2021, pp. 7157–7168.

[47] Y. Yang, A. Loquercio, D. Scaramuzza, and S. Soatto, « Unsupervised moving object detection via contextual information separation », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 879–888.

[48] Y. Yang, B. Lai, and S. Soatto, « DyStaB: Unsupervised object segmentation via dynamic-static bootstrapping », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 2826–2836.

[49] D. Lao, Z. Hu, F. Locatello, Y. Yang, and S. Soatto, *Divided Attention: Unsupervised Multi-Object Discovery with Contextually Separated Slots*, 2023.

[50] X. Xu, L. Zhang, L.-F. Cheong, Z. Li, and C. Zhu, « Learning Clustering for Motion Segmentation », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, *3*, pp. 908–919, 2022.

[51] P. Ochs, J. Malik, and T. Brox, « Segmentation of moving objects by long term video analysis », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, *6*, pp. 1187–1200, 2014.

[52] M. Keuper, B. Andres, and T. Brox, « Motion trajectory segmentation via minimum cost multicuts », *in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, pp. 3271–3279.

[53] A. Kardoost, K. Ho, P. Ochs, and M. Keuper, « Self-supervised sparse to dense motion segmentation », *in Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 - December 4, 2020, Revised Selected Papers, Part II*, H. Ishikawa, C.-L. Liu, T. Pajdla, and J. Shi, Eds., ser. Lecture Notes in Computer Science, vol. 12623, Springer, 2020, pp. 421–437.

[54] S. Wehrwein and R. Szeliski, « Video segmentation with background motion models », *in British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*, BMVA Press, 2017.

[55] E. Meunier, A. Badoual, and P. Bouthemy, « EM-Driven unsupervised learning for efficient motion segmentation », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, *4*, pp. 4462–4473, 2023.

[56] Z. Teed and J. Deng, « RAFT: Recurrent all-pairs field transforms for optical flow », *in Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., ser. Lecture Notes in Computer Science, vol. 12347, Springer, 2020, pp. 402–419.

[57] S. Ghorbani, K. Mahdaviani, A. Thaler, *et al.*, « MoVi: A Large Multipurpose Motion and Video Dataset », *PLOS ONE*, vol. 16, *6*, e0253157, 2021.

[58] A. Geiger, P. Lenz, and R. Urtasun, « Are we ready for autonomous driving? The KITTI vision benchmark suite », *in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, pp. 3354–3361.

[59] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, « A database and evaluation methodology for optical flow », *in IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, IEEE Computer Society, 2007, pp. 1–8.

[60] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, « Monocular Depth Estimation Based On Deep Learning: An Overview », *Science China Technological Sciences*, vol. 63, *9*, pp. 1612–1627, 2020.

[61] M. Irani and P. Anandan, « Parallax geometry of pairs of points for 3D scene analysis », *in Computer Vision — ECCV '96*, vol. 1064, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 17–30.

[62] G. Csurka and P. Bouthemy, « Direct identification of moving objects and background from 2D motion models », *in Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, 566–571 vol.1.

[63] T. Chan and L. Vese, « Active contours without edges », *IEEE Transactions on Image Processing*, vol. 10, *2*, pp. 266–277, 2001.

[64] D. Mumford and J. Shah, « Optimal approximations by piecewise smooth functions and associated variational problems », *Communications on Pure and Applied Mathematics*, vol. 42, *5*, pp. 577–685, 1989.

[65] D. Mahapatra, S. O. Gilani, and M. K. Saini, « Coherency based spatio-temporal saliency detection for video object segmentation », *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, *3*, pp. 454–462, 2014.

[66] Z. Liu, X. Zhang, S. Luo, and O. Le Meur, « Superpixel-based spatiotemporal saliency detection », *IEEE transactions on circuits and systems for video technology*, vol. 24, *9*, pp. 1522–1540, 2014.

[67] Y.-T. Hu, J.-B. Huang, and A. G. Schwing, « Unsupervised video object segmentation using motion saliency-guided spatio-temporal propagation », *in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science, vol. 11205, Springer, 2018, pp. 813–830.

[68] B. Griffin and J. J. Corso, « Tukey-inspired video object segmentation », *in IEEE Winter Conference on Applications of Computer Vision, WACV 2019, Waikoloa Village, HI, USA, January 7-11, 2019*, IEEE, 2019, pp. 1723–1733.

[69] M.-I. Georgescu, A. Barbalau, R. T. Ionescu, F. S. Khan, M. Popescu, and M. Shah, « Anomaly detection in video via self-supervised and multi-task learning », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 12 742–12 752.

[70] R. Chalapathy and S. Chawla, « Deep learning for anomaly detection: A survey », *ArXiv preprint*, vol. abs/1901.03407, 2019.

[71] A. Borji, « Saliency prediction in the deep learning era: Successes and limitations », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, *2*, pp. 679–700, 2021.

[72] L. Jiang, M. Xu, and Z. Wang, « Predicting video saliency with object-to-motion CNN and two-layer convolutional LSTM », *ArXiv preprint*, vol. abs/1709.06316, 2017.

[73] D.-P. Fan, W. Wang, M.-M. Cheng, and J. Shen, « Shifting more attention to video salient object detection », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 8554–8564.

[74] W. Wang, H. Song, S. Zhao, *et al.*, « Learning unsupervised video object segmentation through visual attention », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 3064–3074.

[75] R. Mehran, A. Oyama, and M. Shah, « Abnormal crowd behavior detection using social force model », *in 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 935–942.

[76] B. Solmaz, B. E. Moore, and M. Shah, « Identifying behaviors in crowd scenes using stability analysis for dynamical systems », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, *10*, pp. 2064–2070, 2012.

[77] J.-M. Pérez-Rúa, A. Basset, and P. Bouthemy, « Detection and Localization of Anomalous Motion in Video Sequences from Local Histograms of Labeled Affine Flows », *Frontiers in information and communication technologies*, Computer Image Analysis, 2017.

[78] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, « Crowded scene analysis: A survey », *IEEE transactions on circuits and systems for video technology*, vol. 25, *3*, pp. 367–386, 2014.

[79] B. Zhou, X. Tang, and X. Wang, « Coherent filtering: Detecting coherent motions from crowd clutters », *in Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part II*, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., ser. Lecture Notes in Computer Science, vol. 7573, Springer, 2012, pp. 857–871.

[80] J. Shao, C. C. Loy, and X. Wang, « Scene-independent group profiling in crowd », *in 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, IEEE Computer Society, 2014, pp. 2227–2234.

[81] L. Maczyta, P. Bouthemy, and O. Le Meur, « Trajectory saliency detection using consistency-oriented latent codes from a recurrent auto-encoder », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, *4*, pp. 1724–1738, 2021.

[82] A. P. Dempster, N. M. Laird, and D. B. Rubin, « Maximum likelihood from incomplete data via the EM algorithm », *Journal of the royal statistical society: series B (methodological)*, vol. 39, *1*, pp. 1–22, 1977.

[83] G. Celeux and G. Govaert, « A classification EM algorithm for clustering and two stochastic versions », *Computational Statistics & Data Analysis*, vol. 14, *3*, pp. 315–332, 1992.

[84] V. Badrinarayanan, A. Kendall, and R. Cipolla, « SegNet: A deep convolutional encoder-decoder architecture for image segmentation », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, *12*, pp. 2481–2495, 2017.

[85] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, « Mask R-CNN », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, *2*, pp. 386–397, 2020.

[86] O. Ronneberger, P. Fischer, and T. Brox, « U-net: Convolutional networks for biomedical image segmentation », *in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, N. Navab, J. Hornegger, W. M. W. III, and A. F. Frangi, Eds., ser. Lecture Notes in Computer Science, vol. 9351, Springer, 2015, pp. 234–241.

[87] C. M. Bishop, « Mixture density networks », *Mixture density networks*, 1994.

[88] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, « Video segmentation by tracking many figure-ground segments », *in IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, IEEE Computer Society, 2013, pp. 2192–2199.

[89] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[90] R. J. Hathaway, « Another interpretation of the EM algorithm for mixture distributions », *Statistics & probability letters*, vol. 4, *2*, pp. 53–56, 1986.

[91] J. T. Barron, « A general and adaptive robust loss function », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 4331–4339.

[92] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, « Expectation-maximization attention networks for semantic segmentation », *in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, IEEE, 2019, pp. 9166–9175.

[93] K. Greff, A. Rasmus, M. Berglund, T. H. Hao, H. Valpola, and J. Schmidhuber, « Tagger: Deep unsupervised perceptual grouping », *in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 4484–4492.

[94] K. Greff, S. van Steenkiste, and J. Schmidhuber, « Neural expectation maximization », *in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, *et al.*, Eds., 2017, pp. 6691–6701.

[95] P. Yu, S. Xie, X. Ma, Y. Zhu, Y. N. Wu, and S.-C. Zhu, « Unsupervised foreground extraction via deep region competition », *in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, Virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 14 264–14 279.

[96] V. Monga, Y. Li, and Y. C. Eldar, « Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing », *IEEE Signal Processing Magazine*, vol. 38, *2*, pp. 18–44, 2021.

[97] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, « A naturalistic open source movie for optical flow evaluation », *in Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., ser. Lecture Notes in Computer Science, vol. 7577, Springer, 2012, pp. 611–625.

[98] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, « PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume », *in 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 8934–8943.

[99] L. Liu, J. Zhang, R. He, *et al.*, « Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation », *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, Computer Vision Foundation / IEEE, 2020, pp. 6488–6497.

[100] N. Mayer, E. Ilg, P. Häusser, *et al.*, « A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation », *in 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 4040–4048.

[101] W. Falcon and K. Cho, « A framework for contrastive self-supervised learning and designing a new approach », *ArXiv preprint*, vol. abs/2009.00104, 2020.

[102] D. P. Kingma and J. Ba, « Adam: A method for stochastic optimization », *in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[103] D. C. Liu and J. Nocedal, « On the limited memory BFGS method for large scale optimization », *Mathematical programming*, vol. 45, *1-3*, pp. 503–528, 1989.

[104] A. Papazoglou and V. Ferrari, « Fast object segmentation in unconstrained video », *in IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, IEEE Computer Society, 2013, pp. 1777–1784.

[105] D. Sun, E. B. Sudderth, and M. J. Black, « Layered segmentation and optical flow estimation over time », *in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, IEEE Computer Society, 2012, pp. 1768–1775.

[106] S. Ding, W. Xie, Y. Chen, *et al.*, « Motion-inductive self-supervised object discovery in videos », *ArXiv preprint*, vol. abs/2210.00221, 2022.

[107] B. Duke, A. Ahmed, C. Wolf, P. Aarabi, and G. W. Taylor, « SSTVOS: Sparse spatiotemporal transformers for video object segmentation », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 5912–5921.

[108] S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim, « Video object segmentation using space-time memory networks », *in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, IEEE, 2019, pp. 9225–9234.

[109] C. Ventura, M. Bellver, A. Girbau, A. Salvador, F. Marqués, and X. Giró-i-Nieto, « RVOS: End-to-end recurrent network for video object segmentation », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 5277–5286.

[110] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, *et al.*, « A review on deep learning techniques for video prediction », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, *6*, pp. 2806–2826, 2022.

[111] J. Van Amersfoort, A. Kannan, M. Ranzato, A. Szlam, D. Tran, and S. Chintala, « Transformation-based models of video sequences », *ArXiv preprint*, vol. abs/1701.08435, 2017.

[112] H. Gao, H. Xu, Q.-Z. Cai, R. Wang, F. Yu, and T. Darrell, « Disentangling propagation and generation for video prediction », *in 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, IEEE, 2019, pp. 9005–9014.

[113] C. Finn, I. J. Goodfellow, and S. Levine, « Unsupervised learning for physical interaction through video prediction », *in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 64–72.

[114] P. Luc, C. Couprie, Y. LeCun, and J. Verbeek, « Predicting future instance segmentation by forecasting convolutional features », *in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part IX*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., ser. Lecture Notes in Computer Science, vol. 11213, Springer, 2018, pp. 593–608.

[115] S. S. Nabavi, M. Rochan, and Y. Wang, « Future semantic segmentation with convolutional LSTM », *in British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, BMVA Press, 2018, p. 137.

[116] H. Farazi, J. Nogga, and S. Behnke, « Local frequency domain transformer networks for video prediction », *in 2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–10.

[117] G. K. Batchelor, *An Introduction to Fluid Dynamics*. Cambridge university press, 1967.

[118] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, « The 2017 DAVIS Challenge on Video Object Segmentation », *ArXiv preprint*, vol. abs/1704.00675, 2017.

[119] K. Han, Y. Wang, H. Chen, *et al.*, « A survey on vision transformer », *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, *1*, pp. 87–110, 2023.

[120] F. Locatello, D. Weissenborn, T. Unterthiner, *et al.*, « Object-centric learning with slot attention », *in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M.-F. Balcan, and H.-T. Lin, Eds., 2020.

[121] M. Unser, « Splines: A perfect fit for signal and image processing », *IEEE Signal processing magazine*, vol. 16, *6*, pp. 22–38, 1999.

[122] E. Meunier and P. Bouthemy, « Unsupervised space-time network for temporally-consistent segmentation of multiple motions », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, IEEE, 2023, pp. 22 139–22 148.

[123] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, « Crowded Scene Analysis: A Survey », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, *3*, pp. 367–386, 2015.

[124] L. Maczyta, P. Bouthemy, and O. L. Meur, « CNN-Based temporal detection of motion saliency in videos », *Pattern Recognition Letters*, vol. 128, p. 298, 2019.

[125] M. Guillemot *et al.*, « Breaking Batch Normalization for better explainability of Deep Neural Networks through Layer-Wise Relevance Propagation », *ArXiv preprint*, vol. abs/2002.11018, 2020.

[126] J. Gu, Y. Yang, and V. Tresp, « Understanding individual decisions of CNNs via contrastive backpropagation », *in Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part III*, C. V. Jawahar, H. Li, G. Mori, and K. Schindler, Eds., ser. Lecture Notes in Computer Science, vol. 11363, Springer, 2018, pp. 119–134.

[127] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, « Sanity checks for saliency maps », *in Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 9525–9536.

[128] M. Sundararajan, A. Taly, and Q. Yan, « Axiomatic attribution for deep networks », *in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 3319–3328.

[129] P. Sturmfels, S. Lundberg, and S.-I. Lee, « Visualizing the Impact of Feature Attribution Baselines », *Distill*, vol. 5, *1*, e22, 2020.

[130] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, « Layer-Wise Relevance Propagation: An Overview », *in Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700, Cham: Springer International Publishing, 2019, pp. 193–209.

[131] P. Dabkowski and Y. Gal, « Real time image saliency for black box classifiers », *in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, *et al.*, Eds., 2017, pp. 6967–6976.

[132] A. Zhmoginov, I. Fischer, and M. Sandler, « Information-bottleneck approach to salient region discovery », *in Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part III*, F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, Eds., ser. Lecture Notes in Computer Science, vol. 12459, Springer, 2020, pp. 531–546.

[133] D. P. Kingma and M. Welling, « Auto-encoding variational bayes », *in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[134] J. Yi, E. Kim, S. Kim, and S. Yoon, « Information-Theoretic Visual Explanation for Black-Box Classifiers », *ArXiv*, 2020.

[135] J. Platt and A. Barr, « Constrained Differential Optimization », *in Neural Information Processing Systems*, vol. 0, American Institute of Physics, 1987.

[136] Y. Ganin and V. S. Lempitsky, « Unsupervised domain adaptation by backpropagation », *in Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, F. R. Bach and D. M. Blei,

Eds., ser. JMLR Workshop and Conference Proceedings, vol. 37, JMLR.org, 2015, pp. 1180–1189.

[137] X. Wang, M. Pei, and Z. Nie, « Self-Trained Video Anomaly Detection Based on Teacher-Student Model », *in 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, 2021, pp. 1–6.

[138] X. Wang and A. Gupta, « Unsupervised learning of visual representations using videos », *in Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.

[139] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, « Learning features by watching objects move », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2701–2710.

[140] A. Mahendran, J. Thewlis, and A. Vedaldi, *Cross Pixel Optical Flow Similarity for Self-Supervised Learning*, Jul. 2018.

[141] X. Zhan, X. Pan, Z. Liu, D. Lin, and C. C. Loy, *Self-Supervised Learning via Conditional Motion Propagation*, Apr. 2019.

[142] M. Tangemann, S. Schneider, J. von Kügelgen, *et al.*, « Unsupervised Object Learning via Common Fate », *arXiv:2110.06562 [cs, stat]*, Oct. 2021.

[143] Z. Huang, X. Shi, C. Zhang, *et al.*, « FlowFormer: A transformer architecture for optical flow », *in Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVII*, S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., ser. Lecture Notes in Computer Science, vol. 13677, Springer, 2022, pp. 668–685.

[144] T. Kipf, G. F. Elsayed, A. Mahendran, *et al.*, « Conditional object-centric learning from video », *in International Conference on Learning Representations (ICLR)*, 2022.

[145] R. Steed and A. Caliskan, « Image representations learned with unsupervised pre-training contain human-like biases », *in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 701–713.

[146] I. Pastaltzidis, N. Dimitriou, K. Quezada-Tavarez, *et al.*, « Data augmentation for fairness-aware machine learning: Preventing algorithmic bias in law enforcement systems », *in Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 2302–2314.

[147] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, « EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras », *in Robotics: Science and Systems XIV*, Jun. 2018.

[148] J. Kühne, M. Magno, and L. Benini, « A fast and accurate optical flow camera for resource-constrained edge applications », *arXiv preprint arXiv:2305.13087*, 2023.

[149] S. Kleinman, « Movement notation systems: An introduction », *Quest (Grand Rapids, Mich.)*, vol. 23, *1*, pp. 33–34, 1975.

[150] D. F. Galili, « Gaga: moving beyond technique with ohad naharin in the twenty-first century », *Dance Chronicle*, vol. 38, *3*, pp. 360–392, 2015.

[151] D. P. Kingma, M. Welling, *et al.*, « An introduction to variational autoencoders », *Foundations and Trends® in Machine Learning*, vol. 12, *4*, pp. 307–392, 2019.

[152] M. Chang, T. Griffiths, and S. Levine, « Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation », *in NeurIPS*, 2022.

[153] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, « High Accuracy Optical Flow Estimation Based on a Theory for Warping », *in Computer Vision - ECCV 2004*, ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2004, pp. 25–36.

[154] A. Dosovitskiy, P. Fischer, E. Ilg, *et al.*, « FlowNet: Learning optical flow with convolutional networks », *in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, IEEE Computer Society, 2015, pp. 2758–2766.

[155] A. Ranjan and M. J. Black, « Optical flow estimation using a spatial pyramid network », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 2720–2729.

[156] Z. Zheng, N. Nie, Z. Ling, *et al.*, « DIP: Deep inverse patchmatch for high-resolution optical flow », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, IEEE, 2022, pp. 8915–8924.

[157] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, « FlowNet 2.0: Evolution of optical flow estimation with deep networks », *in 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 1647–1655.

[158] T. Brox, C. Bregler, and J. Malik, « Large displacement optical flow », *in 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 41–48.

[159] S. Jiang, Y. Lu, H. Li, and R. Hartley, « Learning optical flow from a few matches », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 16 592–16 600.

[160] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, « PatchMatch: A randomized correspondence algorithm for structural image editing », *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, *3*, 2009.

[161] S. Meister, J. Hur, and S. Roth, « UnFlow: Unsupervised learning of optical flow with a bidirectional census loss », *in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, S. A. McIlraith and K. Q. Weinberger, Eds., AAAI Press, 2018, pp. 7251–7259.

[162] A. Stone, D. Maurer, A. Ayvaci, A. Angelova, and R. Jonschkowski, « SMURF: Self-teaching multi-frame unsupervised RAFT with full-image warping », *in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, Virtual, June 19-25, 2021*, Computer Vision Foundation / IEEE, 2021, pp. 3887–3896.

[163] V. L. Guen, C. Rambour, and N. Thome, « Complementing brightness constancy with deep networks for optical flow prediction », *in European Conference on Computer Vision*, Springer, 2022, pp. 121–138.

[164] Y. Yang and S. Soatto, « Conditional prior networks for optical flow », *in Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y.

Weiss, Eds., ser. Lecture Notes in Computer Science, vol. 11219, Springer, 2018, pp. 282–298.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Apprentissage non supervisé pour la segmentation et la saillance du mouvement dans des vidéos

Mot clés : segmentation, mouvement, apprentissage non supervisé, flot optique, saillance

Résumé : Les contributions de cette thèse sont de deux ordres. Premièrement, nous avons développé une approche non supervisée d'apprentissage profond pour la segmentation du mouvement à partir du flot optique. Nous avons construit à partir de l'algorithme EM une fonction de perte qui implique des modèles de mouvement paramétriques. Nous avons progressivement ajouté de la cohérence temporelle à cette méthode. Avec un triplet de flots en entrée, nous ajoutons un terme de perte imposant des étiquettes cohérentes au sein du triplet. Ensuite, avec des séquences de flot plus longues en entrée, nous définissons une représentation plus fexible du mouvement par splines, et nous nous appuyons sur un transformer pour appréhender des interactions à long terme entre les caractéristiques. Ces méthodes fournissent des résultats compétitifs sur les benchmarks, tout en étant très efficaces en inférence. La deuxième contribution porte sur la localisation des mouvements saillants à partir du flot optique. Nous supposons que les zones saillantes sont celles qui influencent la prédiction d'un réseau pré-entraîné de classification de saillance. Nous exploitons une méthode d'interprétation du réseau de type gradient pour localiser les zones saillantes. Nous avons également conçu une approche alternative par réseau adverse. Nous avons appliqué ces deux méthodes à deux tâches de saillance du mouvement.

Title: Unsupervised learning for motion segmentation and motion saliency in videos

Keywords: motion segmentation, unsupervised deep learning, optical flow, motion saliency

Abstract: The contributions of this thesis are two-fold. First, we deal with deep learning approaches for fully unsupervised motion segmentation from an optical flow field. We leverage a loss function based on the EM algorithm and involving parametric motion models. We then gradually extend this framework to longer sequences of input flows. With a triplet of input flows, we introduce a loss term enforcing consistent labels within the triplet, and we add long-term temporal consistency with a specific post-processing. Then, we take longer flow sequences as input, and define a spline-based motion representation to handle the evolution of parametric motion over a long time period. In addition, we rely on a transformer decoder to allow interactions between features of the full sequence. These methods provide competitive results on benchmarks, while being very efficient at test time. The second contribution is the localization of salient motions from the optic flow field. In this part, we assume that salients areas are those that influence the output of a pre-trained saliency classification network. We use a gradient-based network interpretation method to localize salient areas. We also design an alternative adversarial approach. We apply both methods on two motion saliency tasks.