Titre: Title:	Apprentissage profond pour vision stéréoscopique multispectrale
Auteur: Author:	David-Alexandre Beaupré
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Beaupré, D.-A. (2020). Apprentissage profond pour vision stéréoscopique multispectrale [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5364/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/5364/
Directeurs de recherche: Advisors:	Guillaume-Alexandre Bilodeau
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Apprentissage profond pour vision stéréoscopique multispectrale

DAVID-ALEXANDRE BEAUPRÉ
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Apprentissage profond pour vision stéréoscopique multispectrale

présenté par **David-Alexandre BEAUPRÉ**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Amal ZOUAQ, présidente
Guillaume-Alexandre BILODEAU, membre et directeur de recherche
Farida CHERIET, membre

REMERCIEMENTS

Je tiens à sincèrement remercier mon directeur de recherche, le professeur Guillaume-Alexandre Bilodeau, pour sa supervision durant l'obtention de ce diplôme ainsi que durant mes stages faits dans les années précédentes. Il était toujours disponible pour discuter et guider mes réflexions lorsque je rencontrais des problèmes, en plus de m'encadrer dans mes activités de recherche depuis plus de trois ans. Je veux aussi remercier le Conseil de Recherches en Sciences Naturelles et Génie du Canada (CRSNG) pour le soutien financier durant ma maîtrise. J'aimerais également profiter de ce moment pour saluer tous mes amis et collègues du laboratoire LITIV que j'ai eu la chance de côtoyer au cours des trois dernières années : Tanushri, Farnoosh, Pierre-Luc, Hui-Lee, Hughes, Soufiane, Zhenxi, Imanne, Pankaj, Alexandre, Yacine, Jules, Zhuofei, Mehdi, Xingfang et Xi. Merci que tout au long de mon parcours, vous m'avez permis d'avoir des discussions enrichissantes, particulièrement lors des dîners. Finalement, je veux remercier les membres du jury qui ont pris le temps d'examiner ce mémoire : professeure Amal Zouaq et professeure Farida Cheriet.

RÉSUMÉ

Ce mémoire présente des méthodes pour estimer les disparités des humains, soit le déplacement entre les pixels des silhouettes humaines, entre des images visibles (RGB) et infrarouges (LWIR). Le but est que, pour chaque pixel dans l'image de gauche, on soit capable de trouver le pixel correspondant dans l'image de droite. Ceci permet de mettre en correspondance les objets d'intérêts d'une scène et peut être utile dans des applications de vidéosurveillance ou de voitures autonomes. Différents facteurs rendent cette tâche plutôt difficile. En plus des difficultés reliées à la nature stéréo du problème, il y a aussi la difficulté de travailler avec deux spectres différents qui n'ont pas beaucoup d'information en commun. Ceci cause beaucoup de problèmes lorsqu'il est temps d'établir des correspondances entre les images. Les méthodes de la littérature se basent sur des descripteurs classiques, mais nous croyons qu'il est possible d'obtenir des méthodes plus performantes si on utilise des réseaux de neurones convolutifs.

Notre projet de recherche présente donc deux nouvelles méthodes basées sur l'apprentissage profond pour estimer des disparités entre des paires d'images multispectrales. La première méthode utilise deux réseaux en parallèle, chacun étant un réseau siamois prenant deux sous-régions en entrées. Pour chaque sous-réseau, le principe est qu'on veut trouver une petite sous-région dans une autre sous-région beaucoup plus large dans l'espace des caractéristiques extraites pour tester différentes translations entre les deux sous-régions. Dans les deux réseaux, on change le domaine des petites et grandes sous-régions, ce qui fait que, dans un cas, on cherche la disparité de RGB dans LWIR, et dans l'autre LWIR dans RGB. Ceci nous permet d'assurer d'avoir des résultats robustes étant donné que la prédiction du réseau se base sur deux résultats. La deuxième méthode est un réseau siamois qui ne partage pas les paramètres entre la branche RGB et la branche LWIR. Les caractéristiques extraites à partir des paires d'images multispectrales sont combinées de deux façons : par corrélation et par concaténation. Utiliser deux manières de combiner les caractéristiques permet au réseau d'être robuste et d'avoir une bonne capacité à généraliser. Des réseaux complètement connectés traitent alors ces nouveaux vecteurs pour donner deux résultats (probabilités) à savoir si les sous-régions d'entrées étaient les mêmes, ou non. Dans les deux cas, les méthodes sont évaluées sur des bases de données publiques où on trouve que la première est compétitive avec les méthodes de la littérature, alors que la deuxième est supérieure aux autres méthodes.

ABSTRACT

This thesis presents new methods to do disparity estimation for human subjects, defined as the distance between pixels on the human silhouettes, between images from the visible (RGB) and infrared domains (LWIR). The goal of disparity estimation is, for each pixel in the left image, to find the corresponding pixel in the right image. This allows the correspondence of objects of interest, which can be useful in applications such as video surveillance and autonomous vehicles. Many factors make this task difficult. It has difficulties related to the stereo aspect of the problem, as well as having to establish correspondences between images from different domains, which is hard since there is not much common information between those. Methods in the literature are based on handcrafted feature descriptors, but we believe that it is possible to obtain better methods if we use convolutional neural networks.

Our research project proposes two new methods based on deep learning to estimate disparity between multispectral image pairs. The first method uses two parallel networks, each one being a siamese network taking two image patches as inputs. For each subnetwork, the idea is that we want to find the location of a small patch inside a larger one in the feature space to test various possible disparities. In both subnetworks, we switch the domains of the small and big patch, so in one case, we find the disparity of RGB in LWIR, and in the other one, LWIR in RGB. This enforces some consistency for the prediction of our network since the results are based on the output of the two subnetworks. The second method is a siamese network that does not share parameters between the RGB branch and the LWIR branch. The extracted features from the multispectral image pairs are combined with two operations: correlation and concatenation. Using two different operations to combine the features allows the network to be more robust and have a good generalization capability. Fully connected layers process the correlated and concatenated vectors to produce probabilities that the inputs were the same or not. Both methods were evaluated on public datasets and we found that the first method is competitive with methods from the literature, while the second one is superior to other methods.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

RNC	Réseau de neurones convolutifs
RGB	Visible
IR	Infrarouge
NIR	Near-Infrared
SWIR	Short-Wavelength Infrared
MWIR	Mid-Wavelength Infrared
LWIR	Long-Wavelength Infrared
FIR	Far-Infrared

CHAPITRE 1 INTRODUCTION

En vision stéréoscopique, le but est d'établir des correspondances entre les pixels de deux images prises avec des caméras positionnées parallèlement et légèrement distancées l'une de l'autre dans une configuration similaire à celle de nos yeux. Ces correspondances permettent entre autres d'estimer la profondeur des objets dans une scène, ce qui est utile pour des applications comme la navigation de robots ou de voitures intelligentes. La plupart des méthodes stéréos travaillent avec deux caméras dans le visible (RGB), mais nous croyons qu'il est avantageux d'utiliser des caméras provenant de spectres différents, comme le visible et l'infrarouge (IR). En travaillant avec une caméra dans l'infrarouge, il est possible de rendre un système stéréoscopique robuste aux variations d'illumination pour la détection d'humains. Par exemple, en vidéosurveillance, la caméra infrarouge est particulièrement utile lorsqu'une personne a un très faible contraste avec son environnement, comme quelqu'un qui marche le soir dans un endroit mal éclairé. La caméra visible aurait de la difficulté à détecter un tel individu, mais une caméra infrarouge n'aurait pas de problème étant donné qu'elle se fie à la température des objets. Par contre, on sait que les caméras infrarouges sont moins performantes que les caméras visibles pour la détection d'humains dans les cas généraux, c'est-à-dire les cas où les contrastes entre le sujet et l'environnement sont élevés. Par conséquent, utiliser des caméras stéréoscopiques de spectres distincts semble être un bon compromis pour avoir un système robuste aux différentes variations d'illumination. L'élément manquant est alors une méthode capable d'estimer la disparité entre les images multispectrales des deux caméras afin de pouvoir mettre en correspondance les mêmes pixels des deux caméras. Ce mémoire s'intéresse donc à développer une méthode d'estimation de disparités pour stéréoscopie multispectrale à l'aide de réseaux de neurones convolutifs (RNCs).

1.1 Définitions et concepts de base

À partir de deux images d'une même scène, il est possible d'estimer la profondeur des objets dans celle-ci. Ce phénomène est le même que l'on retrouve dans le système visuel humain où les deux yeux sont nécessaires pour bien estimer la profondeur des objets que l'on voit. Ce travail de recherche se concentre sur l'estimation de disparités, qui peut être vue comme le mouvement horizontal d'un pixel donné d'une image à l'autre. Plus formellement, si on considère un pixel p aux coordonnées (x_p, y_p) dans l'image de gauche, la disparité d est définie comme le déplacement pour retrouver ce même pixel p dans l'image de droite aux coordonnées $(x_p + d, y_p)$. La figure 1.1 illustre la disparité dans une paire d'images. Les points verts sont

les points qui correspondent entre eux, alors que les points rouges sont les points de l'image de gauche reportés dans l'image de droite sans tenir compte de la disparité. La distance entre les points verts et rouges de l'image de droite est la disparité que l'on cherche à prédire entre les points verts de l'image de gauche et de droite. La profondeur z d'un pixel p de la scène peut alors être calculé par :

$$z = \frac{fB}{d} \tag{1.1}$$

où f représente la longueur focale de la caméra, B la distance entre les deux caméras et d la disparité du pixel p. Nous avons mentionné plus tôt que la disparité est définie comme un déplacement à l'horizontale uniquement. Cette affirmation est vraie tant et aussi longtemps que les images sont rectifiées, ce que nous expliquerons dans les prochains paragraphes.

Figure 1.1 Illustration de la disparité en stéréoscopie : en vert, les points correspondants et en rouge, les points de l'image de gauche projetés dans l'image de droite. La disparité correspond à la distance entre les points verts et rouges dans l'image de droite.

Travailler avec des images rectifiées est essentiel pour obtenir des algorithmes stéréoscopiques efficaces. En effet, la rectification d'images permet de diminuer l'espace de recherche des correspondances de manière significative en transformant la recherche 2D en une recherche 1D. Concrètement, lorsqu'on rectifie les images, on s'assure que pour un pixel p localisé dans l'image de gauche aux coordonnées (x_p, y_p), ce même pixel q dans l'image de droite sera aux coordonnées (x_q, y_q), avec $y_p = y_q$. Par conséquent, il suffit de trouver le bon x_q tel que $x_q = x_p + d$ positionné sur la même ligne dans les deux images.

Nous allons expliquer brièvement comment obtenir des images rectifiées à partir de deux

caméras stéréoscopiques filmant la même scène. La figure 1.2 présente un schéma montrant les éléments essentiels pour comprendre la géométrie épipolaire permettant de rectifier les images. Tout d'abord, notons la présence de plusieurs points d'intérêts. P est un point 3D de la scène observé par les deux caméras. P_G et P_D sont des points 2D de la projection du point P dans les images de gauches et droites. O_G et O_D correspondent à des observateurs du point P dans les caméras de gauche et de droite. E_G et E_D sont les epipôles gauches et droites, soit les points coïncidant entre le plan image et le segment $\overrightarrow{O_G O_D}$ reliant les deux observateurs. Ensuite, on observe trois segments importants. En rouge, on retrouve le segment $\overrightarrow{O_G P}$ passant par le point P_G, en bleu le segment $\overrightarrow{O_D P}$ passant par P_D et en vert, le segment $\overrightarrow{O_G O_D}$ reliant les deux observateurs. Finalement, on peut définir la ligne épipolaire gauche L_G (bleu pointillé) comme le segment reliant les points P_G et E_G (même principe pour la ligne épipolaire droite L_D).

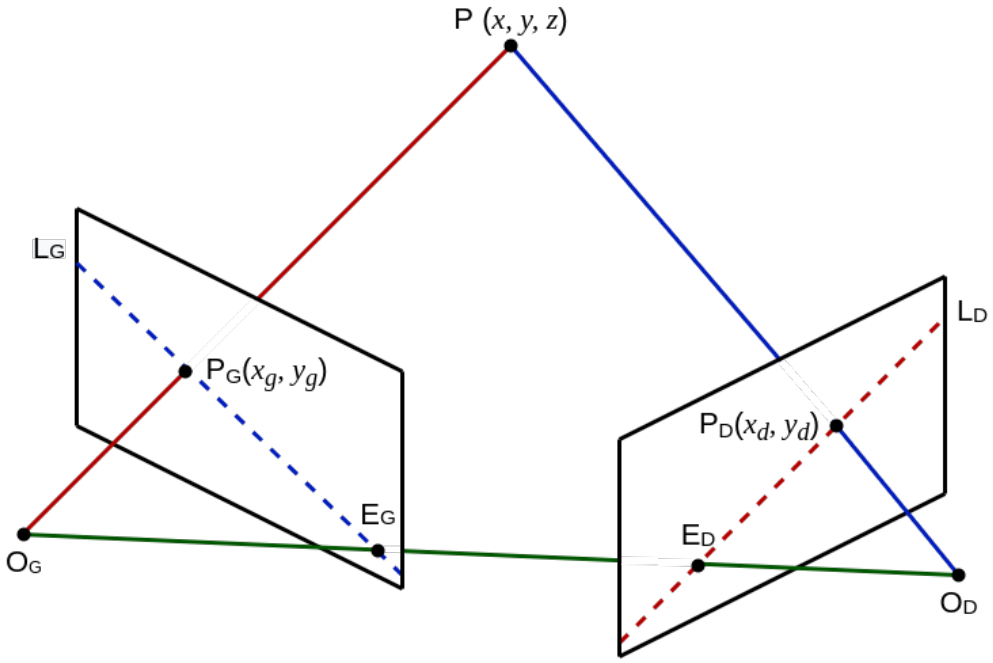


Figure 1.2 Schéma d'un montage stéréoscopique montrant les éléments importants de la géométrie épipolaire.

Une caractéristique intéressante des lignes épipolaires est que pour tout point 3D positionné sur un segment quelconque menant à un point observé, on a la certitude que les projections des points de ce segment dans l'autre image seront sur la ligne épipolaire. Par exemple, dans la figure 1.2, tous les points sur le segment bleu $\overrightarrow{O_D P}$ se retrouve sur le segment L_G en bleu pointillé. Ceci est dû au fait que P, O_G et O_D forment un plan, appelé plan épipolaire, et que les lignes épipolaires sont l'intersection des plans images avec le plan épipolaire. Concrète-

ment, pour un observateur à gauche, il peut y avoir un grand nombre de points sur le segment $\overrightarrow{P_G P}$ en rouge qui auront tous la même projection au point P_G. Or, pour un observateur de l'image de droite, ces points apparaîtront sur la ligne épipolaire L_D étant donné que cette dernière est une projection du segment $\overrightarrow{P_G P}$ dans le plan image de droite. De plus, le fait que la correspondance d'un point donné soit sur une droite nous ramène à un cas de recherche 1D puisqu'on a la certitude que la correspondance du point sera sur la ligne épipolaire.

La dernière étape pour rectifier les images est donc d'aligner les lignes épipolaires des deux images à l'horizontale. Ceci est fait grâce à l'observation qu'un point dans une des images correspond à une ligne épipolaire dans l'autre avec la même matrice fondamentale $\boldsymbol{F}$. On peut définir cette dernière comme suit :

$$L_D = \boldsymbol{F} P_G \tag{1.2}$$

Nous n'irons pas dans les détails sur comment dériver cette équation, mais en multipliant de chaque côté par la transposée du point droit, il est possible de simplifier l'équation 1.2 pour obtenir un système linéaire :

$$P_D^T L_D = P_D^T \boldsymbol{F} P_G \tag{1.3}$$

$$0 = P_D^T \boldsymbol{F} P_G \tag{1.4}$$

Il suffit de résoudre le système linéaire avec au moins sept points P_G et P_D distincts. Maintenant qu'on est capable de calculer $\boldsymbol{F}$, il faut simplement appliquer une homographie sur les deux images afin qu'elles soient rectifiées dans le même plan. On donne un exemple d'image rectifiée avec les lignes épipolaires alignées à la figure 1.3.

Comme plusieurs autres tâches en vision par ordinateur, la vision stéréoscopique a également changé avec l'avènement de l'apprentissage profond. Classiquement, la plupart des algorithmes stéréos suivaient les quatres étapes suivantes :

1. Calcul des coûts de correspondances ;
2. Aggrégation des coûts de correspondances ;
3. Calcul et optimisation des disparités ;
4. Rafinement des disparités ;

L'apprentissage profond, et plus spécifiquement les réseaux de neurones convolutifs (RNCs), a, en premier lieu, changé le calcul des coûts de correspondances en utilisant des caractéristiques profondes au lieu de caractéristiques classiques comme la distance L2. Plus récemment, les quatre étapes ont été remplacées par des architectures bout-à-bout s'occupant de faire toutes ces étapes. Plus de détails sur différentes méthodes de stéréoscopie seront donnés au

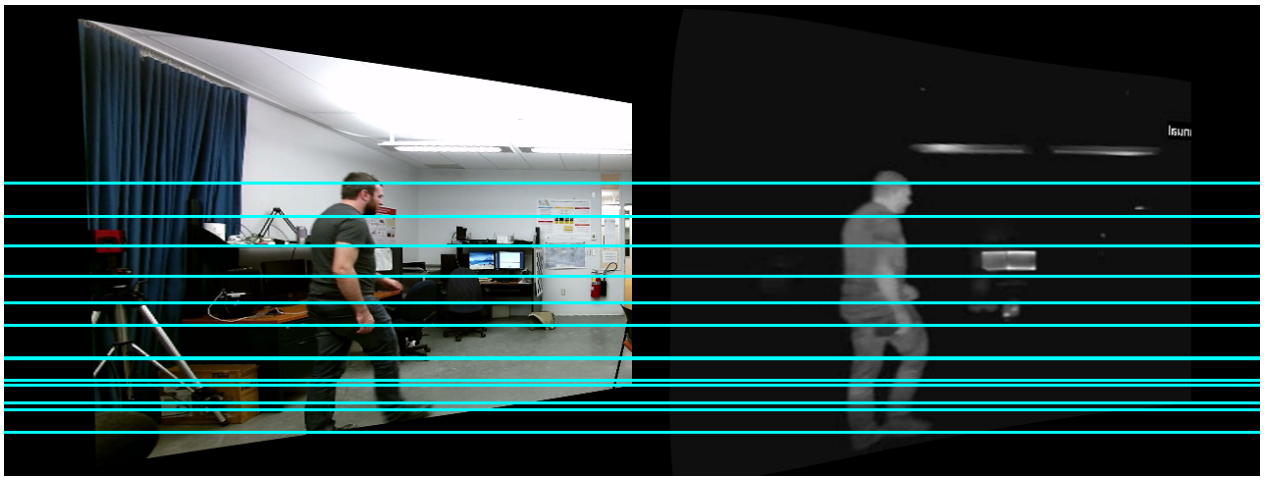


Figure 1.3 Images rectifiées avec les lignes épipolaires alignées.

prochain chapitre.

1.2 Éléments de la problématique

La vision stéréoscopique multispectrale est caractérisée par plusieurs difficultés. Une partie de ces difficultés provient de la nature stéréoscopique du problème, alors que l'autre provient de la nature multispectrale. L'union de ces deux facteurs fait que l'estimation de disparité multispectrale est grandement plus difficile que dans le spectre visible. Les difficultés qu'on retrouve sont les occlusions, les absences de textures et couleurs, les pixels aux frontières, les petits objets et structures minces ainsi que les objets translucides. La figure 1.4 montre des exemples de ces facteurs avec des images de la base de données LITIV 2018 [4].

— **Occlusions** : Un objet visible dans une des images est complètement ou partiellement caché dans l'autre image. Ceci est un des problèmes les plus difficiles à résoudre, étant donné que les pixels de l'objet dans une image n'ont aucune correspondance dans l'autre, ce qui fait qu'il est impossible de prédire leur disparité. Les conséquences de ce problème peuvent être atténuées avec une opération de post-traitement, où les cartes de disparités de gauche et droite sont utilisées pour s'assurer de la consistance des prédictions. Les images (a) et (d) de la figure 1.4 montrent une personne partiellement visible dans l'infrarouge, mais cachée dans le visible.

— **Absence de textures** : Une partie de l'image visible comporte un motif qui se répète, par exemple un motif d'échiquier, et prédire les bonnes disparités pour les pixels du même motif dans l'image infrarouge est un grand défi. Le problème vient du fait que le

motif retrouvé dans l'image de gauche n'est pas visible dans l'image de droite, comme le montrent les images (b) et (d) de la figure 1.4. En effet, le motif à carreaux apparaît comme uni dans l'infrarouge étant donné que l'image mesure la température de la personne, ce qui n'inclut pas le motif de la chemise. Ceci complexifie l'établissement des correspondances.

— **Absence de couleurs** : Similaire au cas de l'absence de textures, l'absence de couleurs en infrarouge rend la recherche de correspondances plus difficile. Les images (c) et (f) de la figure 1.4 montrent bien ce phénomène. En couleur, il est facile de voir la distinction entre les vêtements des sujets, tandis que cette distinction est impossible à voir dans l'infrarouge. L'absence de couleur est encore plus difficile lorsque deux personnes sont près l'une de l'autre étant donné que le nombre de candidats possibles augmente grandement.

— **Pixels aux frontières** : La frontière entre deux objets, ou un objet et son environnement cause des problèmes puisqu'il y a, pour un pixel donné près de la frontière dans l'image de référence, plusieurs candidats possibles dans l'image de correspondance. Ceci fait que sur la carte de disparité, la frontière prédite est parfois un peu décalée par rapport à la frontière réelle. De plus, l'infrarouge peut causer un certain élargissement des frontières étant donné que les objets émettent de la chaleur. Ce phénomène peut être observé dans les images (g) et (j) de la figure 1.4.

— **Structures minces et petits objets** : Cette difficulté est reliée aux pixels de frontières puisqu'il est possible d'argumenter que les petits objets et structures minces ont beaucoup de frontières avec un autre objet ou l'environnement. Ceci fait en sorte que lors des prédictions de disparités, ces objets disparaissent ou leur forme est modifiée. Les images (i) et (l) illustrent ce problème.

— **Objets froids** : Les objets portés ou transportés ayant une température différente du corps humain poseront des difficultés dues à leur changement d'apparence entre le spectre visible et infrarouge. Dans les images (g) et (j) de la figure 1.4, on remarque des objets comme des lunettes qui apparaissent comme noires en infrarouge, alors qu'elles sont translucides dans le visible. Encore une fois, la différence d'apparence des objets entre le visible et l'infrarouge rend la mise en correspondance plus difficile.

Un facteur important en travaillant avec des images infrarouges est le type d'infrarouge qui est utilisé. Il existe cinq types d'infrarouges dans la littérature, soit *Near-Infrared* (NIR), *Short-Wavelength Infrared* (SWIR), *Mid-Wavelength Infrared* (MWIR), *Long-Wavelength Infrared* (LWIR) et *Far-Infrared* (FIR). Parmi ceux-ci, les deux spectres les plus populaires en vision par ordinateur sont le NIR et le LWIR. Ce projet se concentre sur des images LWIR, qui sont très bien adaptées pour la détection des humains. Pour bien comprendre les difficultés des

spectres, nous allons comparer des paires d'images visible-visible, visible-NIR et visible-LWIR que l'on peut observer à la figure 1.5.

— **Visible-visible** : En observant les images (a) et (b) de la figure 1.5, il est très facile de voir que les deux images ont énormément d'information en commun au niveau des textures et couleurs. Les seules différences entre les deux images proviennent du fait qu'elles sont prises de différents points de vue, mais en ce qui a trait à l'apparence des objets, il est évident où ils se trouvent dans les deux images. Par conséquent, les principales difficultés pour une paire d'images comme celle-ci sont les occlusions, les patrons répétés et les surfaces réfléchissantes.

— **Visible-NIR** : Les images (c) et (d) de la figure 1.5 ne proviennent pas d'une base de données stéréoscopique, mais elles montrent tout de même des éléments similaires qui permettent la comparaison des spectres visibles et NIR. Ici, on observe que les différences entre le visible et le NIR sont notables sans être pour autant majeures. Bien qu'il y ait une absence de couleur dans l'image NIR, on remarque la présence de la majorité des textures. On voit que la perte d'information est principalement dans les petits détails, comme les vitraux qui semblent être moins bien définis en NIR. Dans ce type d'infrarouge, l'image ressemble à une image en ton de gris. Ceci fait que les correspondances entre ces types de paires d'images, bien que plus difficiles que visible-visible, sont tout de même faisables dû à une bonne quantité d'information commune entre les deux images.

— **Visible-LWIR** : En regardant les images (e) et (f) de la figure 1.5 on remarque une grande différence entre les informations communes des deux images. En effet, le spectre LWIR est bien adapté pour contraster la silhouette des personnes par rapport à son environnement, au profit des textures et détails. Il n'est pas possible de voir des textures sur les humains de la scène dans ce spectre de l'infrarouge, ce qui rend l'établissement de correspondances entre le visible et LWIR compliqué par rapport aux deux spectres précédents. Une autre difficulté du spectre LWIR est que les frontières des personnes ne sont pas précises lorsqu'on les compare aux frontières dans le visible. Ceci est dû à la nature de l'infrarouge qui mesure la température que nous dégageons, ce qui fais que les prédictions de disparités aux frontières, déjà qu'elles sont problématiques en stéréoscopie visible-visible, le sont encore plus pour de la stéréoscopie visible-LWIR.

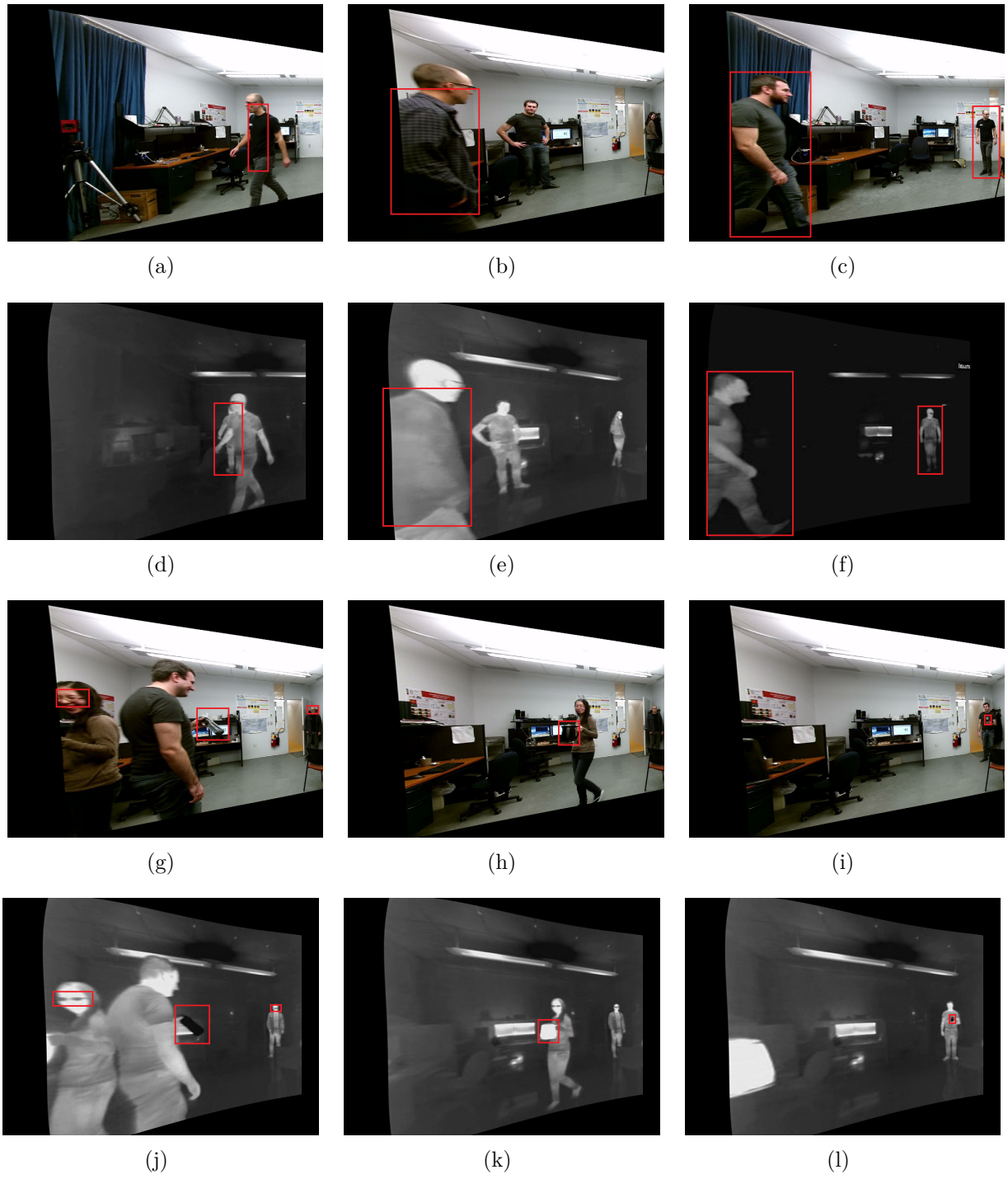
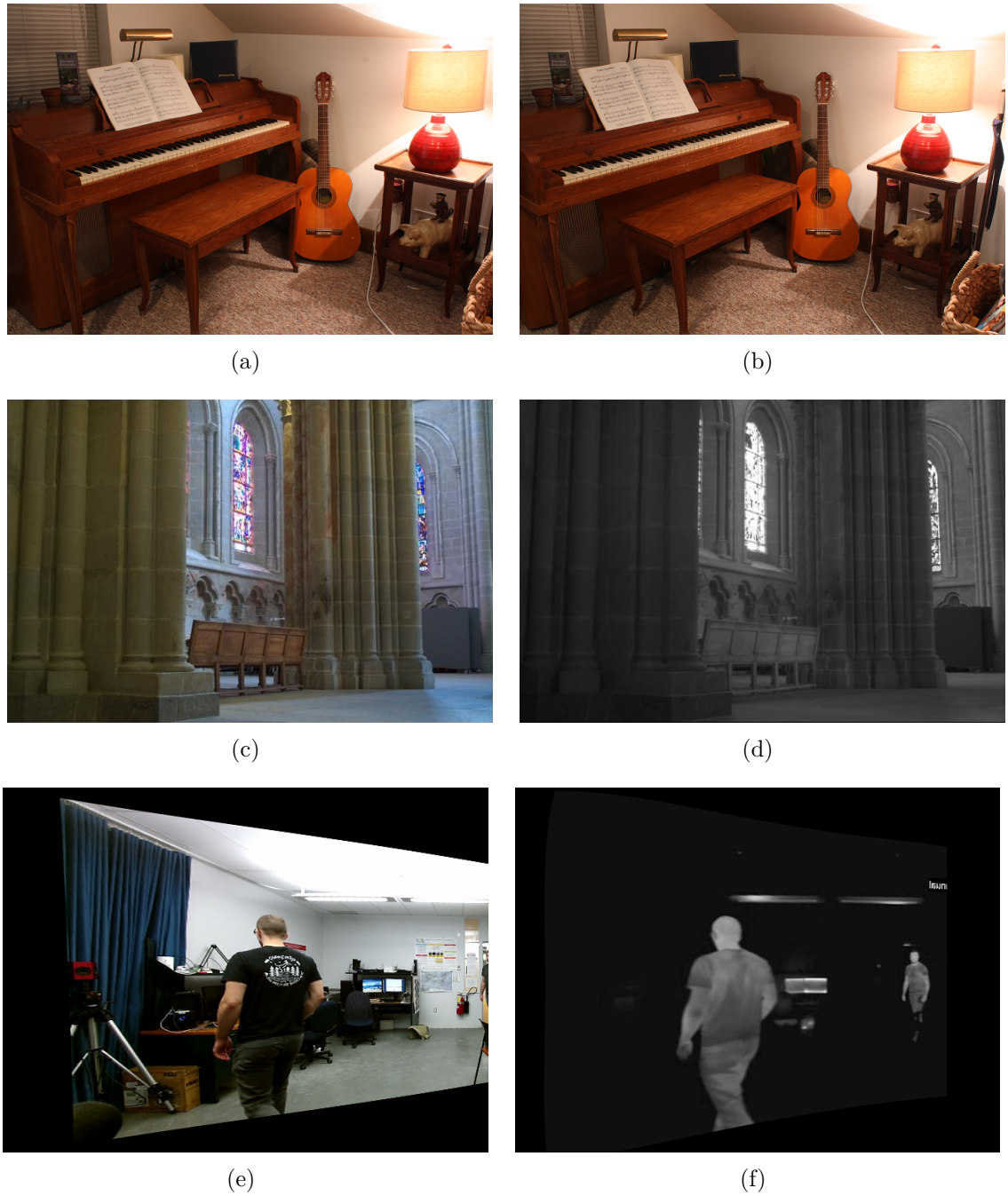


Figure 1.4 Exemples des difficultés en stéréoscopie multispectrale de l'ensemble de données LITIV 2018 [4]. (a)-(d) occlusion, (b)-(e) patrons répétés, (c)-(f) absence de textures, (g)-(j) objets translucides, (h)-(k) pixels aux frontières et (i)-(l) petits objets.



(a) (b)

(c) (d)

(e) (f)

Figure 1.5 Première rangée : images de l'ensemble de données d'images stéréoscopiques Middlebury 2014. Deuxième rangée : images de l'ensemble de données non stéréoscopique VIS-NIR [5]. Troisième rangée : images de l'ensemble de données stéréoscopique multispectral LITIV 2018 [4].

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de produire une méthode capable d'estimer les disparités entre des paires d'images visible et infrarouge. La méthode, basée sur les réseaux de neurones convolutifs (RNCs), doit être robuste aux différents facteurs énoncés à la section 1.2. En détails, les objectifs sont :

— développer de nouvelles architectures de RNCs permettant de décrire des images stéréos multispectrales.
— vérifier que les RNCs sont capables d'obtenir de meilleures performances pour l'estimation de disparité multispectrale par rapport aux descripteurs classiques.
— évaluer les méthodes proposées sur des bases de données publiques et les comparer à d'autres méthodes de la littérature.

1.4 Plan du mémoire

Le chapitre suivant du mémoire présentera une revue de la littérature portant sur des algorithmes classiques de stéréoscopie, des méthodes basées sur l'apprentissage profond, ainsi que des méthodes qui travaillent avec des images multispectrales. Ensuite, au chapitre 3, la démarche générale des méthodes proposées sont présentées, suivi des deux articles de conférences aux chapitres 4 et 5. Ceci est suivi d'une discussion sur les choix que nous avons faits, ainsi qu'une comparaison entre les approches présentées. Finalement, la conclusion propose des améliorations futures ainsi qu'une liste des contributions de ce mémoire.

CHAPITRE 2 REVUE DE LITTÉRATURE

Cette section donnera un aperçu au lecteur de l'évolution des différentes méthodes pour effectuer l'estimation de disparités dans la littérature. De manière générale, les méthodes d'estimation de disparités dans le domaine visible-visible ont de la difficulté avec les occlusions, les régions peu texturées, les régions avec patrons répétés et les surfaces réfléchissantes. On retrouve ces mêmes difficultés pour ce qui est de l'estimation de disparités dans le domaine visible-infrarouge, mais d'autres difficultés s'y ajoutent. Ces dernières ont été abordées en détail au chapitre 1. La principale est le manque d'information commune entre les deux domaines. Puisque l'infrarouge détecte la température des objets, des patrons sur ceux-ci ne sont pas détectés et apparaissent plutôt comme une région uniforme. Par contre, dans le domaine visible, les patrons seront visibles. Faire la correspondance entre ces deux régions est donc difficile, puisqu'il faut apprendre à faire correspondre des régions qui ne sont pas similaires en termes de contenu. Lorsque les deux images sont dans le visible, les patrons sont présents dans les deux images, ce qui facilite grandement les correspondances.

2.1 Méthodes stéréoscopiques classiques

Cette section fera un tour d'horizon des articles en stéréoscopie avant l'avènement de l'apprentissage profond. Étant donné que ce mémoire se concentre sur les méthodes en lien avec les RNCs, nous n'irons pas trop dans les détails dans cette section.

Scharstein *et al.* ont publié une taxonomie [14] des algorithmes stéréoscopiques denses pour permettre de la comparaison de différentes approches. Les auteurs ont proposé quatre étapes que la majorité des algorithmes stéréos effectuent afin de pouvoir effectuer des comparaisons équitables. Les étapes sont :

1. **Calcul des coûts de correspondances** : Plusieurs méthodes comme la différence au carré (SD) [15] ou la différence absolue (AD) [16] sont souvent utilisées pour comparer des pixels. D'autres méthodes basées sur des gaussiennes [17], la corrélation croisée [18] et des caractéristiques binaires [19] sont également étudiées.
2. **Agrégation des coûts** : Des convolutions gaussiennes, des fenêtres glissantes [20] et des fenêtres avec taille adaptable [21] sont tous des options présentées dans la taxonomie. Le but de cette étape est de regrouper des pixels pour créer des zones de recherche dans l'image (appelé souvent volumes de coût).
3. **Calcul et optimisation des disparités** : Ici, on dénote quatre grandes familles :

les méthodes locales (le gagnant ramasse tout (WTA)), les optimisations globales (minimiser une fonction d'énergie [22]), la programmation dynamique (utile pour la stéréo éparse [23]) et les méthodes coopératives (utilisation d'opérations non-linéaires dans des calculs locaux [24]). Ces méthodes sont appliquées sur les zones de recherche de l'étape précédente.

4. **Raffinement des disparités** : Ici, on retrouve des méthodes comme la descente du gradient itérative [15,18], l'utilisation de courbes guides [25], la vérification croisée [26], les filtres médians [27] et autres. L'idée est d'uniformiser les disparités des pixels voisins et réduire le bruit.

Les auteurs ont effectué un grand nombre d'expériences pour évaluer l'effet des différents paramètres à chaque étape. Plusieurs conclusions intéressantes ont été tirées de cet article :

1. Pour des méthodes de calcul de coûts de correspondances comme SD ou AD, la troncation (valeur qui remplace les coûts de correspondance trop grands selon le seuil de troncation) des coûts peut aider les performances dépendant du ratio bruit-signal (SNR).
2. Pour l'agrégation des coûts, les méthodes basées sur des fenêtres ont des difficultés aux changements de profondeur (frontière entre objets), mais performent bien dans les régions texturées.
3. Pour l'optimisation, une comparaison entre la programmation dynamique, l'optimisation sur une ligne de balayage, les coupes de graphe (*graph cut*) et le recuit simulé trouve que les coupes de graphes obtiennent constamment les meilleurs résultats.

L'article de Hirschmuller *et al.* [28] présente la méthode *semi-global matching* (SGM), qui, encore aujourd'hui, est utilisée pour obtenir des cartes de disparités. Cet article présente aussi comment utiliser l'information mutuelle [29] pour former le volume de coût de correspondance, mais SGM peut aussi être utilisé avec d'autres descripteurs comme CENSUS [30] ou HOG [31]. Le principe de SGM est d'agréger les coûts de correspondance 1D selon un certain nombre d'orientations afin de trouver le coût minimal pour chaque pixel à chaque disparité (voir équation 2.1).

$$L_{\boldsymbol{r}}(\boldsymbol{p}, d) = C(\boldsymbol{p}, d) + \min \begin{cases} L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d) \\ L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d-1) + P_1 \\ L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d+1) + P_1 \\ \min_i L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, i) + P_2 \end{cases} - \min_k L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, k) \tag{2.1}$$

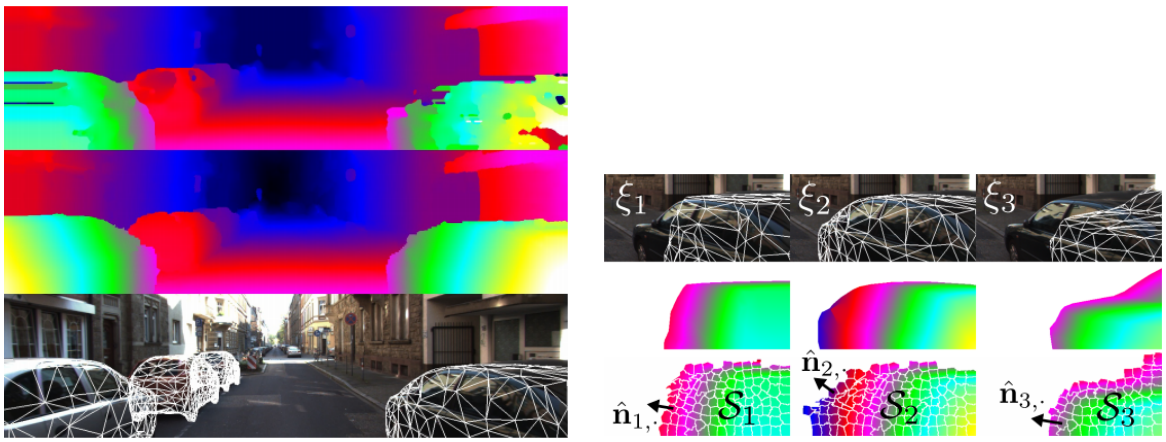
Ici, $L_{\boldsymbol{r}}(\boldsymbol{p}, d)$ représente le coût pour le pixel p à chaque disparité d pour la direction r. On voit

que l'équation 2.1 est définie récursivement, à partir du coût du volume de correspondance $C(\boldsymbol{p}, d)$, du minimum selon la direction r et des pénalités P_1 (pénalité pour les pixels dans le voisinage de $\boldsymbol{p}$) et P_2 (pénalité pour les pixels hors du voisinage de $\boldsymbol{p}$). i et k correspondent à des disparités possibles dans le volume de coûts.

Bleyer *et al.* [32] propose une nouvelle façon de combiner une segmentation sémantique avec l'estimation de disparités. Tous les objets de la scène sont segmentés selon certaines hypothèses : les objets sont compacts en 3D, les objets sont reliés en 3D (il peut par contre y avoir des discontinuités en 2D, dans le cas d'occlusions), un même objet a une apparence relativement uniforme, les objets doivent être raisonnablement gros et les mêmes objets dans les vues de gauches et droites doivent être similaires. Leur méthode consiste à minimiser une fonction d'énergie prenant en compte plusieurs termes comme la cohérence de la photo (pixels similaires ou non d'une vue à l'autre), la cohérence des objets (des pixels voisins devraient appartenir au même objet), la cohérence de la profondeur (dans un petit voisinage, la profondeur devrait être similaire), la couleur des objets (chaque objet de la scène a un modèle de couleur), la parallaxe des objets (les disparités des objets devraient être régulières selon le plan de l'objet), un terme s'assurant que le nombre d'objets dans la scène ne soit pas trop petit (on veut éviter la sur-segmentation) et un terme de connectivité 3D. Cette idée, d'utiliser plus d'une tâche pour améliorer l'estimation de disparités a été reprise récemment dans des méthodes basées sur les RNCs.

Güney *et al.* [7] présente la méthode *Displets* capable d'obtenir des estimations de disparités précises dans les surfaces réfléchissantes, ce qui constitue des régions où les prédictions sont plus difficiles. La figure 2.1 montre comment la méthode est capable de résoudre ce problème. Les *displets* sont définis comme une carte de disparité possible pour un objet sémantique de la scène. Si on prend le cas d'une voiture, les *displets* devraient alors couvrir la forme la plus probable de cet objet. Les *displets* sont aussi utilisés comme une contrainte faible dans la définition des champs aléatoires conditionnels (CRFs). Les CRFs ont démontré d'excellentes performances en optimisation globales pour l'estimation de disparités dans l'article de Scharstein *et al.* [33]. Le CRF utilisé dans cet article est représenté par une fonction d'énergie prenant en compte les quatre éléments suivants :

1. Un terme qui s'assure que les pixels correspondants ont une apparence similaire.
2. Un terme qui encourage la cohérence locale au niveau des superpixels.
3. Un terme qui représente le potentiel unaire (tiens compte de la géométrie à l'intérieur d'une classe sémantique) d'un *displet*.
4. Un terme représentant le potentiel entre chaque *displets* et les superpixels qu'il contient.



(a) (b)

Figure 2.1 (a) Image du haut : carte de disparité de la méthode de Zbontar *et al.* [6]. Image du centre : carte de disparité obtenue par la méthode *Displets* [7]. Image du bas : Géométrie des objets de la scène. Notons une grande amélioration pour l'estimation des disparités dans les fenêtres de véhicules, qui constituent des surfaces réfléchissantes. (b) Illustration d'un *displet*. De haut en bas : image du CAD 3D, suivi du *displet* et de la carte des superpixels avec la normale. De gauche à droite : l'effet d'un changement de CAD 3D sur les *displets*. Les images ont été tirées de [7] ©2015 IEEE & TheCVF.

2.2 Méthodes stéréoscopiques par apprentissage

Cette section abordera l'ensemble des méthodes qui utilisent des RNCs en vision stéréoscopique. La plupart des méthodes présentées proviendront du domaine visible, étant donné que la majorité de la recherche se fait dans ce spectre. Une section avec les méthodes infrarouges sera présentée plus loin. On peut classifier les méthodes par apprentissage en deux grandes familles, soit les méthodes par sous-régions, et les méthodes bout-à-bout. Le principe général dans les deux cas est le même : on extrait des caractéristiques des images de gauches et de droites pour prédire les disparités. Les principales différences proviennent des entrées, ce qui a des conséquences sur les architectures développées. Les prochains paragraphes présenteront les méthodes les plus importantes de chaque catégories. Un concept excessivement important pour cette section est celui du réseau siamois, illustré à la figure 2.2. Son principe est très simple : il s'agit d'un réseau prenant deux entrées et où les paramètres entre des deux branches sont partagés. Ceci fait en sorte que le réseau modifie les paramètres de ses couches de convolutions selon les deux entrées, et non de façon indépendante. Ceci fonctionne très bien dans le cas de la stéréoscopie étant donné que le but est de trouver des régions d'images similaires avec lesquelles on peut établir des correspondances.

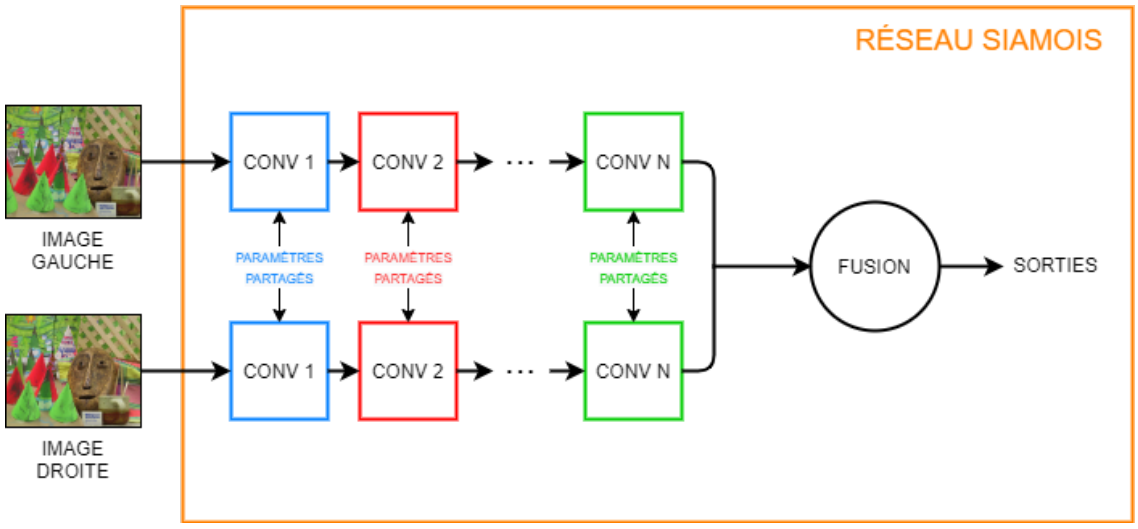


Figure 2.2 Illustration du concept de réseau siamois. Les deux entrées sont passées à deux branches distinctes, où chaque élément dans la branche partage ses paramètres avec son équivalent dans l'autre branche (indiqué par les blocs de même couleur). Une opération de fusion quelconque est utilisée pour joindre les caractéristiques des deux branches.

2.2.1 Méthodes par sous-régions

Ces méthodes sont dites par sous-régions dues à la façon qu'elles sont entraînées. L'entraînement de ces RNCs se fait avec des petites régions d'images provenant des images de gauches et de droites, communément appelées sous-régions. Concrètement, les sous-régions extraites de dimensions $h \times w$ sont centrées sur le pixel p aux coordonnées (x_p, y_p) dans l'image de gauche, et un autre pixel q centré aux coordonnées (x_q, y_q) de l'image de droite. Les avantages de telles méthodes sont multiples. En premier lieu, le fait de travailler avec des entrées de petites tailles réduit la mémoire utilisée par le réseau, ce qui peut être un enjeu dépendant du matériel disponible. De plus, de façon similaire à la mémoire, le temps d'entraînement est également réduit dû à la petite taille des entrées. Finalement, un dernier avantage des sous-régions est que les données n'ont pas besoin d'être annotées de manière dense, c'est-à-dire qu'il n'est pas nécessaire que la majorité des pixels des images aient une disparité connue.

Zbontar *et al.* [6] sont les premiers à avoir utilisé les RNCs pour effectuer de l'estimation de disparités. Leur méthode procédait à l'extraction des caractéristiques provenant de sous-régions 9×9 des images de gauches et de droites. Avec les données de vérité-terrain, les auteurs ont été capables d'établir des paires de sous-régions positives, c'est-à-dire des sous-régions centrées sur des pixels correspondants, ainsi que des sous-régions négatives, soit des sous-régions centrées sur des pixels qui ne correspondent pas. Le but est donc de faire apprendre au réseau à reconnaître quand des paires de sous-régions sont les mêmes, et quand elles sont

différentes, ce qui lui permettra d'estimer les disparités en reconnaissant que deux régions d'images sont similaires. L'architecture proposée, présentée à la figure 2.3 (a), consiste en une couche de convolutions, suivies de couches complètement connectées, une opération de concaténation pour joindre les caractéristiques de gauche et de droite, suivis d'autres couches complètement connectées. La sortie du réseau est un vecteur de deux éléments représentant la probabilité que les entrées soient les mêmes ou non. Il s'agit donc d'une classification binaire. La méthode se base sur l'algorithme de SGM [28] présenté plus tôt pour établir les correspondances denses de la carte de disparité.

La méthode proposée par Chen *et al.* [34] va dans la même direction que la méthode de Zbontar *et al.* [6], mais au lieu d'apprendre si deux sous-régions se ressemblent, ils essayent plutôt de trouver les dissimilarités entre celles-ci. Cette approche considère deux paires de sous-régions de tailles différentes : une petite, et l'autre plus grande pour considérer plus d'information du contexte. La taille des petites patchs est de 13×13 pixels, tandis que la grande est deux fois plus longue et large. Ceci est pour permettre au réseau d'apprendre les différences entre des régions d'images à différentes échelles. L'architecture proposée est deux réseaux siamois, un pour les petites sous-régions, et l'autre pour les grandes sous-régions. Chaque sous-réseau a exactement la même structure, soit quatre couches de convolutions, suivi par une opération de corrélation entre les vecteurs obtenus à partir des sous-régions de gauche et de droite (de même taille). Finalement, les deux sous-réseaux votent pour savoir si les sous-régions en entrées (petites et grandes) sont les mêmes ou non. Tout comme l'article ci-haut, le RNC est utilisé pour construire un volume de coût, qui sera par la suite utilisé par un algorithme comme SGM [28] pour obtenir la carte de disparité finale.

Un inconvénient important de Zbontar *et al.* [6] et Chen *et al.* [34] est que leur méthode prend beaucoup de temps de calcul pour obtenir une carte de disparité. L'article de Luo *et al.* [8] propose une nouvelle approche permettant de réduire le temps d'inférence pour l'estimation des disparités. L'architecture proposée est illustrée à la figure 2.3 (b). Contrairement aux approches précédentes, les deux sous-régions en entrées ne sont pas de la même taille. Plutôt, l'image de gauche est une petite sous-région carrée (taille 9×9), tandis que l'image de droite est une patch beaucoup plus large considérant la disparité maximale, mais de même hauteur (121×9). Le principe de l'architecture est le suivant : les deux sous-régions sont passées dans le RNC siamois pour donner un vecteur de caractéristiques pour la petite sous-région, et un volume de caractéristiques pour la plus grande sous-région. Le volume a la même dimension que le vecteur pour les caractéristiques, mais possède une largeur égale à la disparité maximale déterminée durant l'entraînement. Par la suite, un produit de corrélation est effectué entre le vecteur de caractéristique, et chaque vecteur présent dans le volume caractéristique (corrélation à chaque disparité possible). Ceci donne donc une distribution de

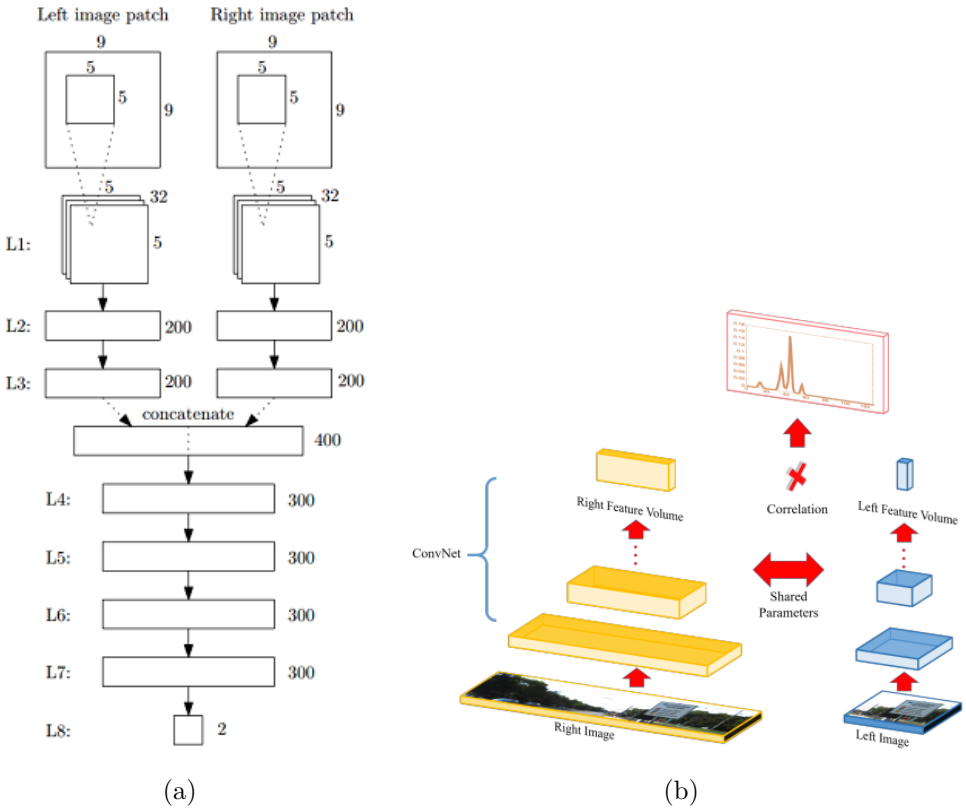
probabilité de trouver le vecteur à une disparité d dans le volume. L'endroit le plus probable est retenu comme la disparité prédite par le réseau. Il s'agit d'une méthode qui effectue de la classification multiclasse, où les classes correspondent aux disparités possibles. De cette manière, cette approche n'utilise pas d'algorithme comme SGM [28], ce qui lui permet de traiter des paires d'images en 0.20 seconde (la méthode de Zbontar *et al.* [6] prenait un peu plus de 20 secondes par paire d'images).

L'article de Park *et al.* [35] montre qu'utiliser des sous-régions plus grandes permet d'obtenir de meilleures caractéristiques pour établir les correspondances entre les régions d'images. Leur raisonnement est que l'humain, lorsqu'il établit des correspondances entre des images, a tendance à regarder tout le contexte d'une région, et non seulement les pixels voisins à l'élément en question. Par conséquent, la méthode de Park *et al.* utilise des tailles de patchs 37×37, soit des sous-régions considérablement plus grande que les approches de Zbontar *et al.* [6] et Luo *et al.* [8]. Ceci va dans le même sens que la méthode de Chen *et al.* [34], où des sous-régions de plus grande taille étaient également utilisées, mais la méthode présente considère encore plus de pixels en entrée. Le principe de leur méthode est très simple : ils utilisent une version modifiée de l'architecture proposée par Zbontar *et al.* [6] où toutes les couches sont convolutives, et ajoutent leur module d'agrégation de pixels en pyramide (4P). Le module 4P est une opération d'agrégation qui augmente le champ réceptif du réseau et est défini comme suit :

$$P^{4P}(\boldsymbol{F}, s) = [P(\boldsymbol{F}, s_1), \ldots, P(\boldsymbol{F}, s_M)] \tag{2.2}$$

$\boldsymbol{F}$ correspond à la carte de caractéristiques et s_i à la taille de l'opération d'agrégation. De cette manière, les caractéristiques à différentes résolutions sont concaténées ensemble, ce qui permet au réseau d'apprendre en considérant plus de pixels, et ce qui permet d'améliorer considérablement les résultats de Zbontar *et al.* [6].

Jusqu'à présent, les méthodes abordées ont principalement deux façons de savoir si les sous-régions en entrées sont les mêmes ou non : effectuer une concaténation des caractéristiques suivie de couches complètement connectées, ou bien faire une corrélation entre les caractéristiques pour savoir si les patchs se ressemblent ou non. Chaque approche a ses avantages, mais l'article de Shaked *et al.* [36] propose une méthode utilisant les deux façons pour améliorer les prédictions de disparités. Les sous-régions de gauches et de droites sont passées à travers un réseau siamois contenant des blocs résiduels inspirés par *ResNet* [37]. Par la suite, les caractéristiques de gauches et de droites sont envoyées dans deux branches distinctes pour la fonction de perte hybride. La première branche va concaténer les caractéristiques de gauche et de droite, et utilisera une fonction de perte d'entropie-croisée binaire pour déterminer si



(a) (b)

Figure 2.3 (a) Architecture proposée par Zbontar *et al.* [6] ©2015 IEEE & TheCVF. (b) architecture proposée par Luo *et al.* [8] ©2016 IEEE & TheCVF. Cette figure illustre la différence entre la concaténation (a) et la corrélation (b) pour joindre les informations des images de gauche et droite, ainsi que la différence des tailles des sous-régions en (b).

les sous-régions sont les mêmes ou non. Ceci est similaire à ce qui est fait par Zbontar *et al.* [6]. Dans l'autre branche, les caractéristiques de gauche et de droite sont corrélées avant la fonction de perte de Hinge, ce qui donne une idée de la similarité des sous-régions. Cette branche se rapproche de ce qui est fait par Luo *et al.* [8]. Ces deux branches forment donc une fonction de perte hybride où deux objectifs doivent être atteints parallèlement, ce qui permet au réseau de trouver des caractéristiques plus robustes, et ce qui se traduit par de meilleurs résultats lors de l'estimation de disparités. Contrairement aux méthodes précédentes, les auteurs entraînent un autre réseau pour obtenir les cartes de disparités finales.

2.2.2 Méthodes bout-à-bout

Les méthodes bout-à-bout diffèrent des méthodes par sous-régions puisqu'elles ne requièrent qu'un seul pipeline d'entraînement pour obtenir des cartes de disparités. Les méthodes abordées à la section précédente (par sous-régions) étaient souvent utilisées pour former un volume

de coût de correspondances, qui était ensuite fourni à un algorithme comme SGM [28]. Dans le cas des méthodes bout-à-bout, c'est le réseau qui apprend, à partir du volume de coût, à prédire les disparités. Par conséquent, au lieu d'avoir des sous-régions en entrées, ces méthodes prennent les images de gauches et de droites dans leur entièreté. Les méthodes bout-à-bout ont l'avantage d'être plus précises que les méthodes par sous-régions, et généralement les disparités aux frontières sont beaucoup plus lisses étant donné que le réseau apprend à faire l'optimisation sur l'ensemble de l'image. Par contre, elles nécessitent beaucoup de plus de mémoire, et bien que le temps d'inférence est comparable, le temps d'entraînement peut être excessivement long. Dans le cas des méthodes bout-à-bout, il y a aussi une restriction du côté des annotations disponibles étant donné que celles-ci doivent être denses ($\geq 50\%$ des pixels ont une disparité connue).

Le réseau *DispNet* proposé par Mayer *et al.* [38] a été le premier à utiliser une architecture bout-à-bout. Avant cet article, un problème qui empêchait le développement de réseaux bout-à-bout était le manque de données pour entraîner ces réseaux contenant beaucoup de paramètres. Les ensembles de données existants étaient trop petits pour entraîner les architectures sans qu'il y ait du sur-apprentissage. Mayer *et al.* propose donc un nouvel ensemble de données synthétiques où chaque pixel possède une disparité connue. Cet ensemble de données sera utilisé par toutes les méthodes qui seront présentées subséquemment. Le réseau qu'ils proposent est très simple : il s'agit d'une réseau avec deux phases, soit une phase de compression, suivi d'une phase de décompression, inspiré de l'architecture FlowNet [39]. Le but de la phase de compression est de trouver les caractéristiques les plus significatives des images d'entrées en réduisant la taille spatiale des cartes de caractéristiques. De son côté, la phase de décompression doit construire une carte de disparité à partir des caractéristiques en augmentant progressivement la résolution jusqu'à l'obtention de la résolution originale.

Une approche similaire à *DispNet* [38] est celle proposée par Pang *et al.* [40], intitulée CRL (*Cascade Residual Learning*). Il s'agit d'une architecture en deux étapes, où la première reprend l'architecture de *DispNet* pour obtenir une carte de disparités. La contribution de cet article se retrouve dans la deuxième étape. Celle-ci consiste en un réseau de raffinement de disparités utilisant les images de gauches et de droite, une image de gauche transformée grâce à la carte de disparité de l'étape 1, et une carte d'erreur entre l'image de gauche originale et celle transformée. L'architecture de la deuxième étape a aussi une phase de compression et décompression, mais son rôle est d'améliorer les prédictions de la carte de la première étape dans les zones difficiles (occlusions, patrons répétés, régions sans textures). Le réseau de raffinement permet d'améliorer les résultats de *DispNet*.

L'approche proposée par Kendall *et al.* [9] est la première qui met l'emphase sur l'importance

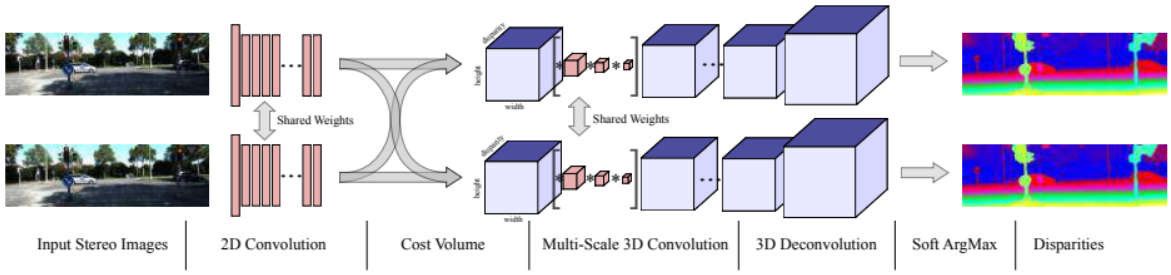


Figure 2.4 Architecture de *GC-Net* permettant d'estimer des disparités par Kendall *et al.* [9] ©2017 IEEE & TheCVF. La plupart des architectures ont un pipeline similaire : une phase d'extraction des caractéristiques, formation du volume de coût, optimisation/raffinement du volume de coût et prédiction de la carte de disparités.

de la géométrie du contexte pour les architectures bout-à-bout. Ce qui est sous-entendu par géométrie et contexte est que, par exemple, une voiture est un objet rigide ayant une certaine forme (géométrie) et qu'une fenêtre fait partie d'un objet comme une voiture ou un bâtiment (contexte). L'architecture *GC-Net*, présentée à la figure 2.4, contient des convolutions 3D qui se charge de considérer le contexte de la scène, en plus de régulariser le volume de coût des correspondances. Une autre contribution de l'article est de considérer la tâche comme une tâche de régression, au lieu d'une classification. Plusieurs travaux subséquents considéreront aussi le problème comme une régression étant donné qu'il est facile d'aller cherche une très bonne précision avec une telle formulation.

Dans la même veine que *GC-Net* [9], Chang *et al.* [41] propose une nouvelle architecture pour améliorer les disparités dans les régions difficiles. Ceci est également fait en considérant le contexte. Pour ce faire, les auteurs utilisent une agrégation spatiale en pyramide (SPP) [42] pour obtenir des caractéristiques à plusieurs niveaux. Le module SPP est placé juste avant le volume de coût, ce qui fait que ce dernier tient compte des détails de l'image à plusieurs niveaux. Pour ce qui est de la régularisation du volume de coût, une structure en sablier (phase de compression suivie d'une de décompression) est utilisée. Celle-ci est répétée à trois reprises, et après chaque décompression, il y a une régression pour obtenir une carte de disparité. Le réseau est donc entraîné avec trois cartes de disparités, chacune ayant une précision variable (la dernière prédit mieux les détails que la première).

Tel que présenté plus tôt, SGM [28] est un algorithme stéréo extrêmement populaire dû à sa simplicité et sa bonne vitesse d'exécution. Seki *et al.* [43] ont appris les paramètres P_1 et P_2, responsables de régulariser les changements de disparités aux frontières, à l'aide d'un processus d'entraînement itératif selon les différentes directions d'agrégation. Plus récemment, *GA-Net*, proposé par Zhang *et al.* [44] utilise une version modifiée des équations de SGM [28]

pour définir deux nouvelles couches d'un réseau de neurones. La couche d'agrégation locale (LGA) est utilisée pour traiter l'information locale (arêtes et structures minces) et est définie comme suit :

$$C^A(\boldsymbol{p}, d) = somme \begin{cases} \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} \omega_0(\boldsymbol{p}, \boldsymbol{q}) \cdot C(\boldsymbol{q}, d) \\ \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} \omega_1(\boldsymbol{p}, \boldsymbol{q}) \cdot C(\boldsymbol{q}, d-1) \\ \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}}} \omega_2(\boldsymbol{p}, \boldsymbol{q}) \cdot C(\boldsymbol{q}, d+1) \end{cases} \tag{2.3}$$

$$t.q. \sum_{\boldsymbol{q} \in N_{\boldsymbol{q}}} \omega_0(\boldsymbol{p}, \boldsymbol{q}) + \omega_1(\boldsymbol{p}, \boldsymbol{q}) + \omega_2(\boldsymbol{p}, \boldsymbol{q}) = 1$$

$\boldsymbol{p}$ est un pixel à la position (x, y), $\boldsymbol{q}$ un autre pixel dans le voisinage N_p de $\boldsymbol{p}$, d la disparité, ω_i les poids à apprendre pour la couche LGA et $C(\boldsymbol{q}, d)$ le coût de correspondance entre $\boldsymbol{p}$ et $\boldsymbol{q}$ à la disparité d. $C^A(\boldsymbol{p}, d)$ est le coût d'agrégation local. Ce dernier est utilisé dans l'équation 2.4 qui est utilisé pour définir SGM [28] de manière compatible avec la rétropropagation :

$$C_{\boldsymbol{r}}^A(\boldsymbol{p}, d) = C(\boldsymbol{p}, d) + somme \begin{cases} \boldsymbol{w}_1(\boldsymbol{p}, \boldsymbol{r}) \cdot C_{\boldsymbol{r}}^A(\boldsymbol{p} - \boldsymbol{r}, d) \\ \boldsymbol{w}_2(\boldsymbol{p}, \boldsymbol{r}) \cdot C_{\boldsymbol{r}}^A(\boldsymbol{p} - \boldsymbol{r}, d-1) \\ \boldsymbol{w}_3(\boldsymbol{p}, \boldsymbol{r}) \cdot C_{\boldsymbol{r}}^A(\boldsymbol{p} - \boldsymbol{r}, d+1) \\ \boldsymbol{w}_4(\boldsymbol{p}, \boldsymbol{r}) \cdot \max_i C_{\boldsymbol{r}}^A(\boldsymbol{p} - \boldsymbol{r}, i) \end{cases} \tag{2.4}$$

Cette équation est de la même forme que l'équation 2.1 présentée plus tôt. Les principales différences sont les poids ω_i appris par le réseau, l'utilisation d'une somme au lieu d'un minimum et le changement du minimum pour un maximum dans le quatrième terme de la somme. Cette dernière modification est due à la formulation de la fonction de coût par les auteurs, qui préféraient maximiser les probabilités au lieu de les minimiser.

La méthode proposée par Guo *et al.* [45] explore deux opérations de fusion populaires pour joindre les caractéristiques des images de gauche et droite : la corrélation et la concaténation. La corrélation a pour avantage d'être rapide, mais contrairement à la concaténation, elle perd beaucoup d'information (la dimension des canaux est perdue). L'architecture proposée utilise donc deux volumes de coût, soit un obtenu par la concaténation des caractéristiques, et l'autre obtenu par la corrélation de caractéristiques groupées. La principale contribution de l'article se trouve dans la formation du volume de coût groupé. Si on détermine que les caractéristiques $\boldsymbol{f}_l$ ont N_c canaux, et qu'on veut N_g groupes, alors nous devrons grouper les caractéristiques dans des groupes de $\frac{N_c}{N_g}$ éléments. Une opération de corrélation sera alors faite sur ces groupes de caractéristiques, et les caractéristiques résultantes seront concaténées

pour former le volume de coût groupé. L'équation suivante montre comment la corrélation groupée est effectuée :

$$\boldsymbol{C}_{gwc}(d,x,y,g) = \frac{1}{N_c/N_g}\langle \boldsymbol{f}_l(x,y), \boldsymbol{f}_r(x-d,y)\rangle \tag{2.5}$$

Les expériences faites par les auteurs ont montré que d'utiliser deux volumes de coûts, un par concaténation et l'autre par corrélation groupée, permet d'augmenter les performances par rapport à lorsqu'un seul est utilisé.

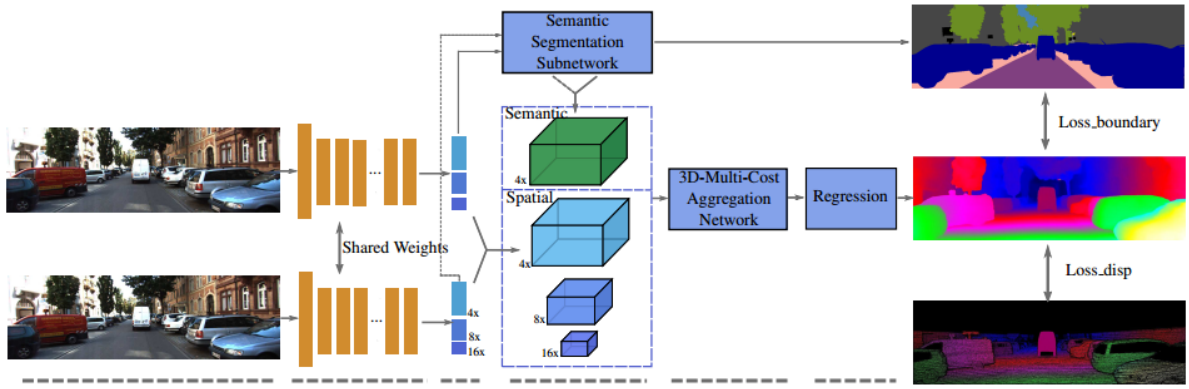


Figure 2.5 Architecure proposée par Wu *et al.* [10] ©2019 IEEE & TheCVF. Les autres méthodes utilisant une autre tâche pour aider à l'estimation de disparité suivent le même principe : une branche s'occupe de la disparité, et une autre pour la tâche connexe. C'est la fonction de perte qui se charge de s'assurer que le réseau optimise les deux tâches en même temps.

Dernièrement, plusieurs méthodes de stéréoscopie tentent de résoudre une autre tâche connexe pour améliorer les estimations de disparités. Xioa *et al.* [46] utilise une carte des arêtes des objets de la scène pour aider la prédiction des disparités aux frontières. Le raisonnement est que les changements de disparité arrivent le plus souvent sur une frontière entre deux objets, donc utiliser la carte des arêtes permet de guider le RNC vers ses régions problématiques. Yang *et al.* [47] propose de leur côté d'utiliser une carte de segmentation sémantique pour aider l'estimation de disparités. Le réseau est entraîné de manière non-supervisée pour la partie de segmentation et supervisé pour les disparités. Tout comme pour les arêtes, la segmentation sémantique peut être bénéfique pour les pixels aux frontières, mais aussi pour les surfaces uniformes avec très peu de textures. Wu *et al.* [10] vont plus loin avec la segmentation sémantique en utilisant le concept du SPP de Chang *et al.* [41] pour obtenir des caractéristiques du contexte à différentes résolutions. Par contre, ils utilisent le principe des pyramides au niveau des volumes de coûts, et non avant la formation de celui-ci. Ceci permet

au réseau d'avoir plus d'information pour la partie d'estimation de disparité étant donné qu'il a plusieurs volumes de coûts. La figure 2.5 montre l'architecture proposée par les auteurs. Le fait d'utiliser plus d'une tâche pour entraîner le réseau permet d'améliorer sa robustesse, sa capacité à généraliser et les prédictions puisqu'il peut utiliser plusieurs indices pour prédire les disparités.

2.3 Méthodes multispectrales

Cette section présentera différents travaux qui travaillent avec des images multispectrales. Il est important de noter que certains de ces travaux ne sont pas en LWIR, mais plutôt en NIR. Il est tout de même intéressant d'étudier ces méthodes puisqu'elles peuvent avoir des idées qui s'appliquent bien au domaine LWIR. De plus, certains de ces travaux ne sont pas en stéréoscopie, mais ils utilisent deux images de la même scène où le but est d'établir des correspondances entre les pixels de ces images. Une fois de plus, nous jugeons que ces travaux sont pertinents par rapport au sujet de ce mémoire.

Bilodeau *et al.* [1] présente une étude comparative de plusieurs descripteurs classiques pour l'estimation de disparités entre des images RGB et LWIR. Parmi ces descripteurs, on retrouve l'information mutuelle [29], la somme des différences au carré (SSD) [1], le descripteur SIFT [48], le descripteur HOG [31], le descripteur LSS [49], le descripteur binaire FREAK [50] et le descripteur binaire BRIEF [51]. La conclusion principale de cet article est que globalement, l'information mutuelle obtient les meilleures performances, mais que le descripteur *LSS* est supérieur lorsque les régions sont de plus petite taille.

Baruch *et al.* [52] propose une nouvelle méthode pour effectuer la détection et la correspondance de points clés dans des paires d'images RGB et LWIR. Leur méthode est un RNC hybride étant donné qu'il utilise deux sous-réseaux pour établir les correspondances. Le premier est un réseau siamois classique tel que présenté à la figure 2.2. Le deuxième est un réseau siamois, sans le partage des paramètres entre les deux branches. Ceci signifie que le réseau possède des paramètres pour l'image dans le visible et pour l'image dans l'infrarouge, au lieu de les partager. Ceci est une proposition intéressante des auteurs, puisqu'ils montrent que de garder les deux spectres séparés dans un des RNCs permet d'obtenir de meilleures performances pour ce qui est de la correspondance multispectrale des points clés.

Pistarelli *et al.* [53] reconnaît que la principale difficulté de travailler avec des images RGB et LWIR est le manque d'information en commun entre les deux domaines. Pour pallier à ce problème, ils proposent une méthode se basant sur les arêtes uniquement pour établir leurs correspondances. Ils soutiennent que ceux-ci seront des éléments communs retrouvés dans

les deux spectres, donc ils utilisent l'algorithme Canny [54] pour détecter les arêtes. Par la suite, ils effectuent une transformation pour les mettre dans un espace commun, soit l'espace de Hough. Les arêtes dans ce nouvel espace sont considérées comme des caractéristiques qui doivent trouver leur correspondance dans l'autre espace de Hough, ce qui se fait en vérifiant les candidats les plus similaires le long de la ligne épipolaire.

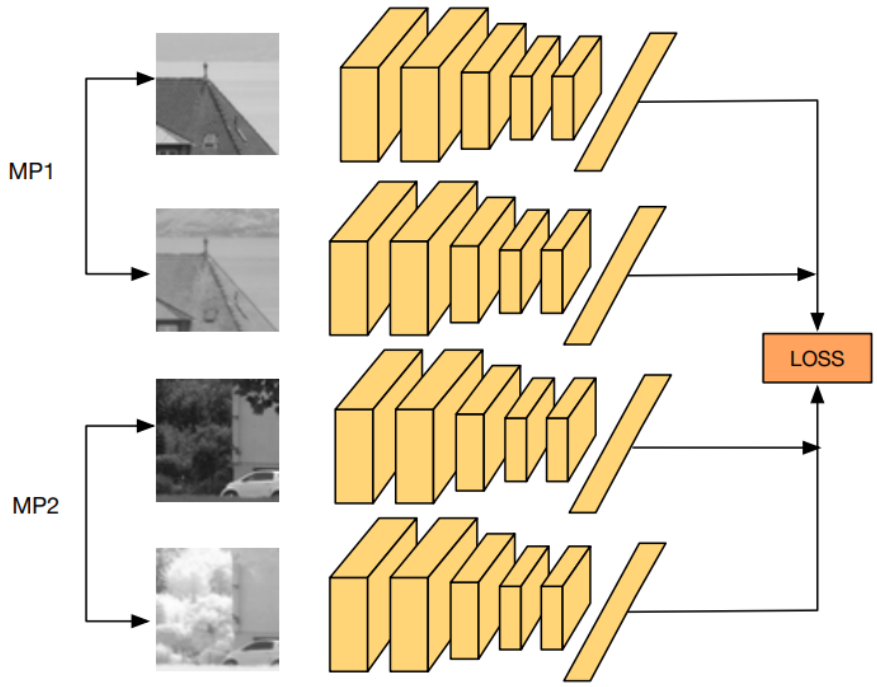


Figure 2.6 Présentation du réseau quadruplet [11] ©2017 MDPI Sensors. Le réseau prend en entrée deux paires d'images correspondantes, mais différentes entre elles.

Aguilera *et al.* [55] propose un descripteur basé sur les histogrammes d'arêtes orientées. La première étape consiste à utiliser SIFT [48] pour détecter des points-clés, qui seront ensuite décrits selon des histogrammes. Ceci est fait dans les images RGB et LWIR. Par la suite, ils mettent en correspondance les points en minimisant la distance euclidienne entre leurs descripteurs. Dans un travail subséquent, Aguilera *et al.* [56] comparent trois différentes architectures de RNCs pour déterminer si les réseaux de neurones sont capables d'apprendre des caractéristiques plus riches que les descripteurs classiques. Ce travail est effectué sur des paires d'images RGB et NIR, où les similitudes entre les images sont moins grandes que dans le visible. Ce travail trouve qu'un architecture à deux canaux, soit un RNC non siamois où les images sont concaténées pour être l'entrée du réseau donne les meilleurs résultats. Dans un autre article, Aguilera *et al.* [11] propose une nouvelle architecture pour établir des correspondances entre les spectres RGB et NIR. Il s'agit d'un réseau quadruplet, prenant en entrée

quatre images : une image RGB avec une image NIR correspondante, et une autre image RGB et NIR correspondante, mais différente de la première paire. L'architecture est montrée à la figure 2.6. De cette façon, le réseau a deux exemples positifs (les paires correspondantes), et quatre exemples négatifs (les images RGBs qui ne correspondent pas avec l'autre image NIR, et les paires d'images RGB-RGB et NIR-NIR). La formation de ces six éléments donne un objectif clair au réseau : il faut trouver les caractéristiques communes aux paires d'images, mais celles-ci doivent aussi être discriminantes par rapport aux autres images.

Les approches de Zhi *et al.* [12] et Liang *et al.* [57] entraînent des réseaux capables d'estimer les disparités de manière non-supervisée entre des images RGB et NIR étant donné le manque d'annotations denses dans la vision multispectrale. Le principe des deux méthodes est similaire, soit celui de créer des pseudos images de gauches et de droites pour les deux spectres. Par exemple, si l'image de gauche est dans RGB et l'image de droite dans NIR, alors ces méthodes créeront une pseudo image de gauche dans NIR et une pseudo image de droite dans RGB. De cette façon, il est plus facile d'établir des correspondances dans les mêmes spectres. L'approche de Zhi *et al.* [12] utilise un réseau à plusieurs étapes pour prédire les disparités, créer les pseudo-images et effectuer la comparaison entre les images originales et créées pour la fonction de coût. Ils utilisent aussi une segmentation de certains matériaux (vitre, vêtements, végétation) pour améliorer la qualité des pseudo-images. Les différents modules de leur approche sont présentés à la figure 2.7. La méthode de Liang *et al.* [57] utilise un *CycleGAN* [58] modifié pour créer les pseudo-images. Par la suite, les paires d'images de même spectre sont passées dans un RNC qui prédit une carte de disparité pour chaque paire d'images. Cette carte de disparité est alors fournie au *CycleGAN* pour créer des pseudo-images plus réalistes, ce qui permet au réseau d'apprendre.

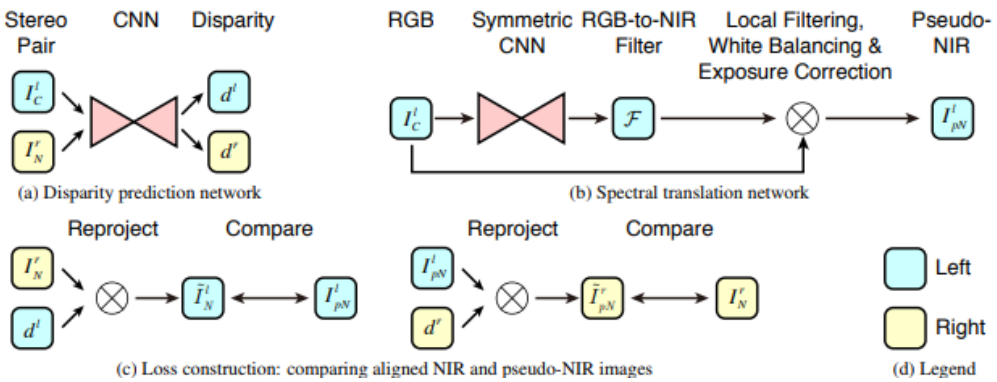


Figure 2.7 (a) Réseau pour la prédiction des disparités. (b) Réseau qui crée les pseudo-images dans les deux spectres. (c) Définition de la fonction de perte en comparant les images originales et celles créées. Image tirée de [12] ©2018 IEEE & TheCVF

St-Charles *et al.* [4] propose une méthode pour segmenter et prédire les disparités entre des paires d'images RGB et LWIR. La méthode consiste à utiliser deux fonctions d'énergie qui doivent être minimisées : une pour la segmentation binaire, et l'autre pour la stéréoscopie. La fonction d'énergie stéréoscopique prend en compte l'apparence, la forme, l'unicité et la régularité. De son côté, l'énergie de segmentation tient compte de la couleur, des contours, de la régularité et d'un terme qui s'assure de la cohérence temporelle. Ce processus est fait itérativement, ce qui fait que la segmentation aide la stéréoscopie, qui elle-même aide à son tour la segmentation multispectrale.

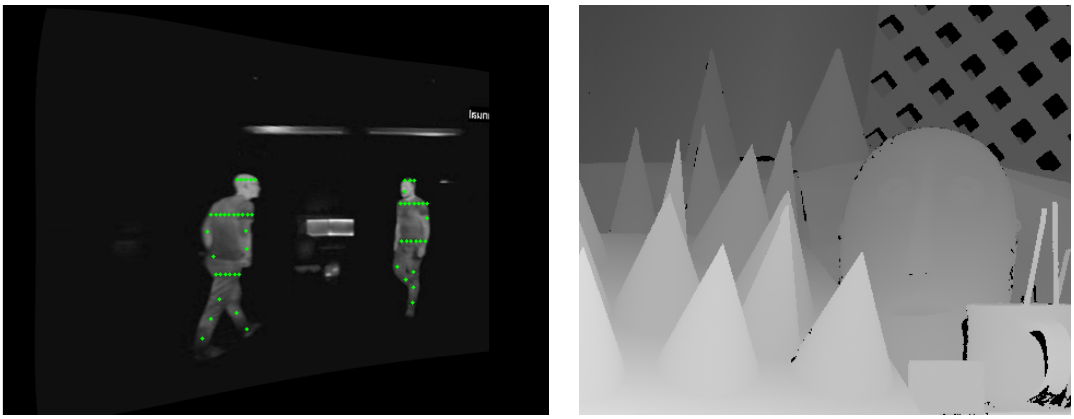
2.4 Synthèse

Ce chapitre a présenté différentes méthodes par rapport à la stéréoscopie visible et multispectrale. Nous avons discuté de certaines méthodes plus classiques qui sont moins utilisées depuis l'avènement des méthodes basées sur l'apprentissage profond. Nous avons vu que les méthodes par sous-régions avaient l'avantage de ne pas nécessiter des annotations denses et permettaient un entraînement rapide, mais que leurs cartes de disparités manquaient de précision. De leur côté, les méthodes bout-à-bout sont celles qui obtiennent les meilleures performances, au coût de réseaux de très grande taille nécessitant beaucoup de matériel et de temps d'entraînement. D'un autre côté, il y a de plus en plus d'intérêt pour des travaux avec de l'infrarouge, mais il s'agit d'une tâche plus difficile due, entre autres, à un manque de bases de données, ce qui motive le développement de méthodes nécessitant peu ou pas de supervision.

CHAPITRE 3 DÉMARCHE DE L'ENSEMBLE DU TRAVAIL DE RECHERCHE

Les différentes méthodes présentées au chapitre 2 nous ont permis de développer deux architectures distinctes pour prédire les disparités entre des images multispectrales RGB et LWIR. Cette section détaillera le processus de développement des deux méthodes.

Un choix commun pour les deux articles provient d'une contrainte des ensembles de données utilisées. Ce choix est que les architectures développées seront des méthodes par sous-régions puisque les données multispectrales à notre disposition sont annotées de façon éparse. La figure 3.1 présente un exemple d'annotations des bases de données utilisées pour ce projet de recherche, et compare celles-ci avec des annotations qualifiées de denses provenant du domaine visible. On voit très facilement que le nombre de points annotés n'est pas comparable, ce qui fait que les deux architectures que nous avons développées seront par sous-régions étant donné qu'il est plus facile de les entraîner lorsqu'il y a moins de données disponibles par image. C'était également un problème pour la stéréoscopie RGB-RGB avant la création de l'ensemble de données *Sceneflow* [59] alors que la plupart des méthodes étaient aussi par sous-régions. Une autre difficulté par rapport aux ensembles de données utilisés est, étant donné qu'il n'y a pas énormément d'annotations, il est plus difficile d'entraîner un réseau qui ne sur-apprend pas. Nous avons utilisé des techniques d'augmentation de données pour fournir plus de points à nos méthodes. Celles-ci seront expliquées en détail dans les articles.



(a) (b)

Figure 3.1 Différence d'annotations denses d'un ensemble de données (LITIV 2018 [4] utilisés dans cette recherche (a). Exemple d'annotations denses de l'ensemble de données Middlebury [13] (b).

Pour le premier article, le but était de déterminer s'il était possible pour un RNC d'être compétitif pour l'estimation de disparités multispectrales par rapport aux descripteurs classiques qui obtenaient de bons résultats (information mutuelle et LSS [1]). Pour ce faire, nous nous sommes inspirés de l'architecture proposée par Luo *et al.* [8] présentée à la section précédente. Le principe de notre méthode est d'extraire des caractéristiques des sous-régions RGB et LWIR, pour ensuite trouver, dans l'espace des caractéristiques, l'endroit le plus probable de la petite sous-région RGB dans la grande sous-région LWIR. Un des aspects de notre méthode est qu'on travaille avec deux domaines d'image différents, donc les spectres de la petite et de la grande sous-région peuvent avoir une influence sur les performances de l'architecture. Pour améliorer les performances, nous nous sommes inspirés de la méthode de Chen *et al.* [34] où deux branches du même réseau étaient utilisés pour considérer des sous-régions de tailles différentes. Or, dans notre cas, nous utilisons deux branches du même réseau pour, d'un côté, trouver la petite sous-région RGB dans la grande LWIR, et de l'autre, trouver la petite sous-région LWIR dans la grande RGB. De cette façon, nous avons un réseau plus robuste qui peut être plus confiant dans ses décisions lorsque les prédictions des deux branches s'accordent. Dans le cas où les branches donnent des disparités différentes, on peut simplement se baser sur celle qui a le maximum le plus clair. À la figure 3.2, on démontre un cas où deux branches sont mieux qu'une seule. Si on avait seulement la branche du réseau de gauche N_L, la prédiction serait erronée. Si on avait seulement celle du réseau de droite N_R, la prédiction serait correcte. Or, si on utilise les deux branches en additionnant les distributions de probabilités, on obtient toujours la bonne prédiction, étant donné que le maximum reste au même endroit grâce à la distribution de N_R.

Pour le deuxième article, le but était de développer une architecture supérieure aux descripteurs classiques. Nous avons fait différent du premier article étant donné que les résultats étaient limités. Tout d'abord, nous avons changé l'architecture pour considérer des sous-régions d'entrées de même taille. Les articles de Shaked *et al.* [36] et Guo *et al.* [45] ont influencé certains de nos choix. Tout d'abord, nous commençons par extraire les caractéristiques des sous-régions RGB et LWIR pour obtenir des vecteurs décrivant les deux sous-régions. Contrairement au premier article, le réseau siamois utilisé ne partage pas ses paramètres entre la branche RGB et la branche LWIR. Ceci a été influencé par l'article de Baruch *et al.* [52]. Les vecteurs sont alors joints de deux façons différentes : par corrélation, et par concaténation. Utiliser à la fois la corrélation et la concaténation est une idée provenant de Shaked *et al.* [36] et Guo *et al.* [45]. Dans le cas de Shaked *et al.* [36], deux branches, une obtenue par concaténation et l'autre par corrélation, sont utilisées pour calculer deux fonctions de coûts. Du côté de Guo *et al.* [45], la corrélation et la concaténation sont utilisées pour former deux volumes de coûts distincts. De façon similaire à ces deux cas, nous avons

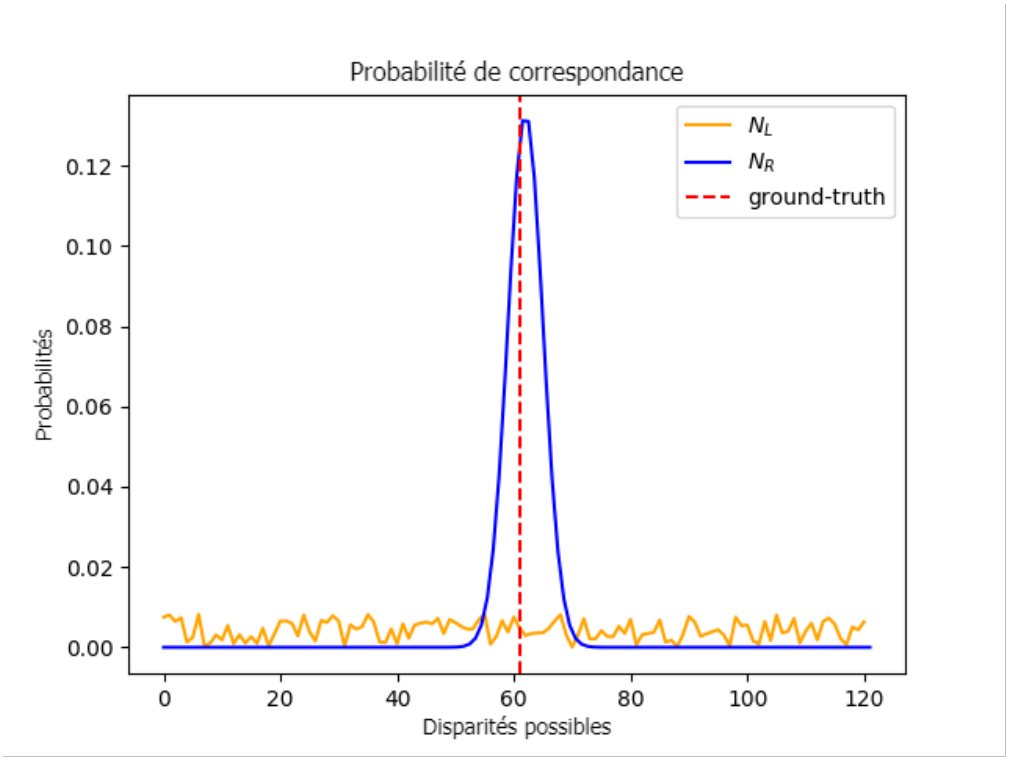


Figure 3.2 Cas où la prédiction du réseau de gauche (N_L) est plutôt uniforme, alors que celle du réseau de droite (N_R) est claire. Avec les deux branches, on s'assure d'une bonne prédiction, ce qui n'est pas nécessairement le cas avec une seule branche.

donc des vecteurs de caractéristiques de corrélation et d'autres de concaténation. Chaque vecteur est alors propagé dans une couche complètement connectée pour donner une probabilité que les sous-régions d'entrées soient les mêmes ou non. Pour prédire les disparités, on utilise une régression similaire à ce qu'il est fait dans l'article de Kendall *et al.* [9]. Pour ce faire, on utilise une sous-région RGB de même taille que lors de l'entraînement, et une sous-région LWIR plus large selon la disparité maximale. On extrait les caractéristiques des deux sous-régions pour obtenir un vecteur pour RGB et un volume pour LWIR. Pour chaque disparité possible dans le volume de caractéristique LWIR, on fait la corrélation et la concaténation avec le vecteur LWIR à la disparité d du volume pour obtenir une probabilité que ces deux vecteurs (provenant d'une sous-région centré sur un pixel) correspondent. Lorsque ceci est fait pour toutes les disparités, on effectuer la somme pondérée des disparités par les probabilités normalisées obtenues. Un exemple simplifié à la figure 3.3 illustre ce processus.

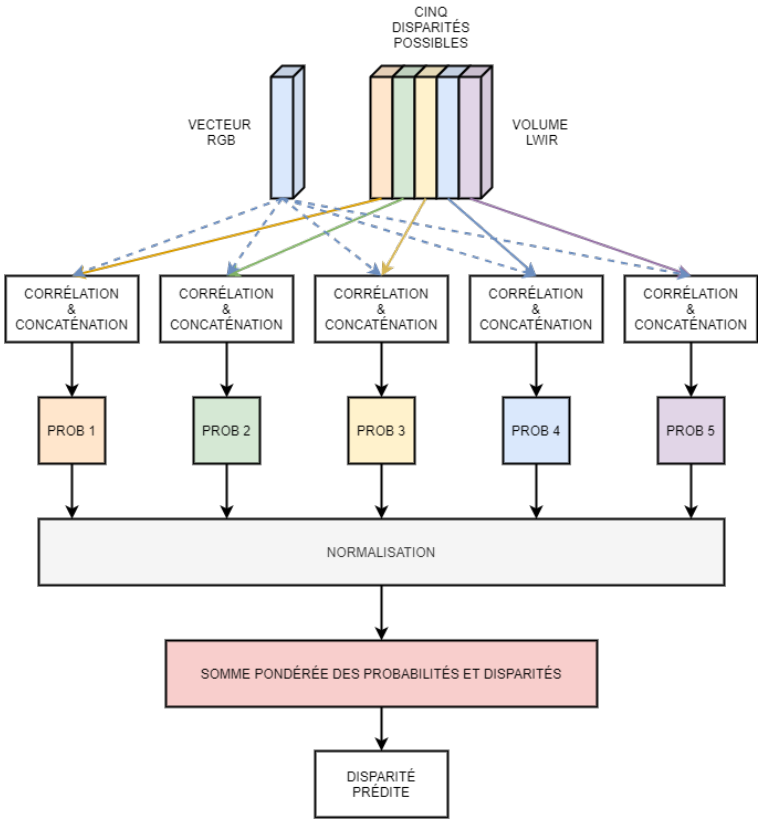


Figure 3.3 Processus d'inférence du deuxième article à partir d'un exemple simplifié. Dans le volume LWIR, les positions de disparités vont de zéro à la valeur de disparité maximale, un paramètre choisi à l'inférence.

CHAPITRE 4 ARTICLE 1 : SIAMESE CNNs FOR RGB-LWIR DISPARITY ESTIMATION

D.-A. Beaupre and G.-A. Bilodeau,
publié à The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019

Abstract

Currently, for the task of color (RGB) and thermal infrared (LWIR) disparity estimation, handcrafted feature descriptors such as mutual information are the methods achieving best performance. In this work, we aim to assess if convolutional neural networks (CNNs) can achieve competitive performance in this task. We developed an architecture made of two subnetworks, each consisting of the same siamese network, but taking different image patches as input. Each siamese network, in the feature space, searches for the disparity between the left and right patch. The output of the two subnetworks are summed together so that we can be more confident in the predicted disparity by enforcing left-right consistency. We show that having two subnetworks working together in parallel to get the final prediction helps achieve better performance when compared to a single subnetwork by itself. We tested our method on the LITIV dataset and found the results competitive when compared to handcrafted feature descriptors. The source code of our method will be available online[1] *upon publication.*

4.1 Introduction

Stereo matching remains one of the most fundamental task in the field of computer vision. With it, we can register stereo images from two different cameras into the same coordinate system from which operations such as depth estimation and object detection can be performed. Recently, more work with stereo image pairs outside of the visible domain has been done. Having one of the images in the LWIR (Long Wavelength Infrared or thermal infrared) spectrum can help with some problems affecting a stereo image pair in the visible spectrum. For example, a person wearing dark clothes at night will be more difficult to perceive in the visible spectrum than in the infrared spectrum. Anytime a person has a low color contrast with its environment, detection in the visible domain will be challenging. However, in the thermal infrared domain, this will be much easier because we use the heat of the person's

1. Available at `https://github.com/beaupreda`

body to do the detection. In the LWIR domain, detections do not rely on textures or colors, but simply on the person having a different temperature from its environment. Nonetheless, this restricts the class of objects that we can work with *i.e.* we cannot work with objects that do not emit heat. For example, stereo matching is a method widely used in self-driving car and the usage of an infrared camera, in this case, would not be always helpful as road signs and urban furniture often do not emit heat. This is because objects of interest need to have a different temperature than the ambient temperature in order to be detected. So, in the case of humans, working in the infrared domain makes sense because our body temperature, in most cases, is warmer than the ambient temperature when working with indoor or outdoor videos. Thermal images, while not perfect, seem well adapted for detecting humans and their combination with visible images can be beneficial. This is why the proposed model is targeted to do disparity estimation for human silhouettes.

Recently, convolutional neural networks (CNNs) have shown to perform very well in multiple tasks of computer vision such as object detection [60, 61], classification [37, 62, 63], tracking [64,65] and stereo matching [8,9,34,36,40,41,66,67]. Most of the recent works in deep learning for stereo matching is, however, mostly in the visible domain, where we know that CNNs are able to extract meaningful features in order to do matching. In the thermal infrared domain, there is less textures and intensity differences, therefore the task of stereo matching between the visible and thermal domain is harder [68]. This work aims at investigating if the recent successes of deep learning in visible stereo matching can be translated to visible-thermal stereo matching. More precisely, we want to assess if a method that learns descriptors outperform the ones using handcrafted feature descriptors that are presently the state-of-the-art for visible-infrared image pairs. Classical methods to match pixels from two different spectrum (visible and thermal) often consist of using descriptors (SIFT [48], HOG [31], mutual information [29] and others) and then using a sliding a window approach to match the features between the images.

We developed a new model, consisting of two subnetworks working in parallel, where each subnetwork receives a pair of multispectral stereo image patches as input. The domains of these inputs are inverted for each subnetwork. Each input pair is made of a small square patch and a larger patch. The stereo pair image patches are forwarded into a CNN module (siamese network) where the goal is to find the location of the small patch inside the bigger one. This is done through a siamese network made of convolutions, batch normalization [2] and ReLU activations [3], with the two branches being merged with a correlation layer to output a log-probability vector for each possible disparity location. Since we have two subnetworks working in parallel, we have two vectors of log-probabilities (one that predicts the disparity of RGB inside LWIR, and the other that predicts LWIR inside RGB). We sum those two

log-probabilities vector and take the index of the maximum value to get our final disparity prediction. The goal of having two subnetworks is that they enforce some sort of consistency by learning to give the same disparity prediction. In summary, our main contributions are:

— We propose a new model made of two subnetworks, each of them taking multispectral stereo pair image patches as input. The outputs are log-probabilities vector which when summed, give a disparity prediction between the visible and the infrared domain. Our model also enforces left-right consistency for the disparity predictions.

— We perform experiments on the LITIV dataset [1] to validate the performance of our model. These results show that a CNN is able to be competitive with handcrafted feature descriptors.

The organization of the paper is the following: in section 4.2, we discuss related work. In section 4.3, we present our proposed model. In section 4.4, we present the dataset used to evaluate our method, some data augmentation that we did and our results. Finally, in section 4.5, we conclude this paper.

4.2 Related Work

In the task of disparity estimation for human silhouettes between the visible and the thermal infrared domain, handcrafted feature descriptors are still the preferred approach. There are three families for categorizing similarity measures. The first category consists of methods that compute similarity across all the pixels inside a given window. Methods such as mutual information [29] fall into this category. Mutual information [29] computes statistics of co-occurrence of intensities for all pixels of two given windows. Because it is able to find a match between a textured region and an uniform one, this method gives good result for disparity estimation between two different domains. Another method falling into this category would be Sum of Squared Differences (SSD) [1] consisting of summing the squared difference of each pixels in two windows, one in each image of an image pair. The second category includes methods that model data as distributions. This category incorporates methods [1, 49, 69] that rely on descriptors such as LSS [70], SIFT [48] and HOG [31]. LSS is a local descriptor able to capture self-similarity among colors, textures and repetitive patterns, which makes it, like mutual information, proficient at matching textured regions with uniformed ones. SIFT is different from LSS because it is based on gradients. SIFT uses interest points that are invariant to illumination, rotation, viewpoint and scale to match objects from two different images or windows. For HOG, since it is already based on windows, we simply need to measure the distance between two histograms coming from two different windows to get a similarity score. The third category measures the binary comparison of pixels and include

methods [1] based on FREAK [50] and BRIEF [51]. Both of these features compute a binary vector representing each window to match and with the hamming distance, the similarity between the windows is obtained. These methods have the advantage of being faster than SIFT.

In the task of stereo matching in the visible domain, deep networks are the methods that achieve state-of-the-art results. We can separate deep learning methods into two categories: the ones that are patch-based and the ones doing end-to-end learning. In the patch based methods, *Zbontar et al.* [67] were the first to show that it was possible to use a CNN to do stereo matching. Their method consisted of taking two small patches from a stereo image pair and with a CNN, classifying if the two patches were a good match or a bad match. *Luo et al.* [8] takes one small patch from the left image and a larger one from the right image and treats the problem as a multi-class classification, where the different classes are all possible disparity values. They also join the features of their siamese network with a inner product which produces very good results in term of computation. Our siamese network showed in section 4.3.2 is inspired by this previous work. *Jie et al.* [66] used constant highway networks [36] to produce a stereo matching cost volume and then used it as input inside a recurrent neural network (RNN) to do the disparity estimation. The main idea was to learn the left-right consistency verification during training and utilizing the error maps given by that verification as attention module to guide the network towards those areas during training. Constant highway networks [36], as mentioned above, produce a matching cost volume based on both inner and outer residual shortcuts. They also introduce a way to output a disparity map and a confidence of said map with the creation of a global disparity network featuring a reflective confidence. For the end-to-end methods, *Kendall et al.* [9] said that many problems in stereo matching could be solved by using geometry *i.e.* knowledge of the environment around the matching points. They proposed using 3-D convolutions to incorporate context and act as a regularizer over the cost volume. They also show that treating the problem of stereo matching as regression gives better performance than treating it as a classification problem. *Chang et al.* [41] also tries to take advantage from context information. In order to do this, they introduced a spatial pyramid pooling module to get different scales *i.e.* generate feature maps of different sizes and also introduced a stacked hourglass (encoder-decoder architecture) to get more information from the context. *Pang et al.* [40] proposed an end-to-end cascade method composed of two stages. The first stage is responsible of producing fine-grained disparities with an up-convolution module and is the input of the second stage where the disparities are rectified with residual signals.

Our method is based on patches because densely annotated visible-thermal infrared datasets are not available.

4.3 Method

This section presents the proposed model made of two subnetworks working in parallel, each being a siamese network, to achieve the final disparity prediction. The global architecture can be viewed in figure 4.1.

4.3.1 Network Architecture

Our network is composed of two siamese networks taking a total of four different inputs to achieve a final disparity prediction y for a given location. We will refer to each subnetwork (siamese network branch) as N_L and N_R for the left and right subnetworks respectively. There are two inputs for each subnetwork, the left input being a small square patch of size $p_s \times p_s$ and the right input being much larger, but of the same height, of size $p_r \times p_s$ where p_s and p_r are defined as follows:

$$p_s = 2 \times p_{hs} + 1 \tag{4.1}$$

$$p_r = 2 \times p_{hr} + p_s + 1 \tag{4.2}$$

where p_{hs} and p_{hr} are hyperparameters of the network, where they represent half the size and range of a patch. These two values will be discussed in section 4.4.1. The subnetwork N_L has two patches as inputs, $P_{N_L}^{RGB}$ and $P_{N_L}^{LWIR}$, the first one being in the visible domain (RGB) and the other one being from the infrared domain (LWIR). For the N_L branch, the patch $P_{N_L}^{RGB}$ is the left input while $P_{N_L}^{LWIR}$ is the right input. For the other subnetwork (N_R), the domains of the inputs are simply inverted, so the left input is $P_{N_R}^{LWIR}$ and the right one is $P_{N_R}^{RGB}$. Basically, we try to find the location of an RGB patch inside an LWIR patch in subnetwork N_L, while in N_R, we try to find the location of an LWIR patch inside an RGB patch. N_L and N_R share all their parameters.

Each subnetwork, composed of a siamese network module (see figure 4.2, outputs a vector of log-probabilities, $\log(p_L)$ and $\log(p_R)$ for the left and right branches respectively, of size p_r. These vectors represent the probability of finding the center of the left patch at disparity location d inside the larger right patch. We can then define the prediction of the left and right branch as y_{N_L} and y_{N_R} respectively. Instead of working with only one of the two branches and taking the maximum of either p_L or p_R as the prediction, we sum those two vectors together and take the position j of the maximal value as the prediction for the disparity. Formally we can then define the final prediction y as follows:

$$y = \arg\max_j(\log(p_L) + \log(p_R)) \tag{4.3}$$

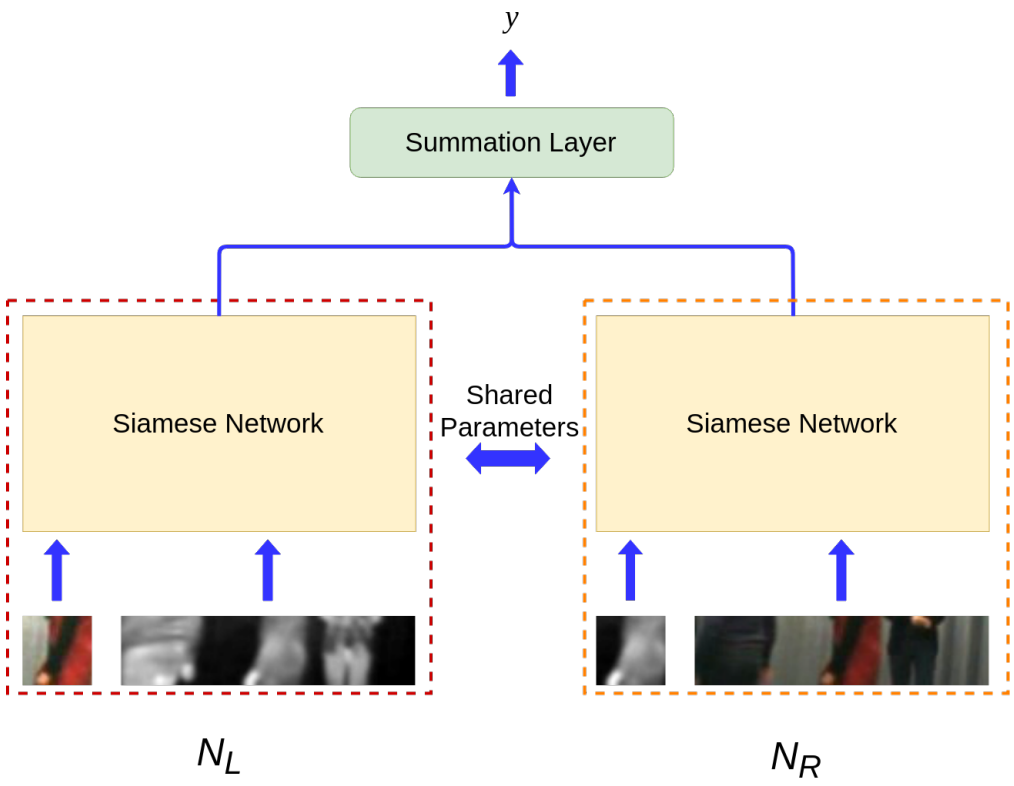


Figure 4.1 Architecture of our proposed model showcasing the two subnetworks N_L and N_R.

This is done for a number of reasons. First of all, adding the predictions of both subnetworks N_L and N_R enforces some sort of left-right consistency where the two subnetworks will work together to learn to give the same disparity prediction y_L and y_R. If both branches consistently give the same output, then we can be more confident in the said prediction. Second of all, there are cases where one of the two subnetworks give more information than the other one. For instance, let us say that the distribution of p_L is mostly uniform and therefore, there are no clear disparity location where a maximum arises. If we were to only use this branch for our final prediction y, we would obtain incorrect predictions most of the time. However, in a case like this one, the distribution of p_R might have a clear maximum at a correct disparity location, meaning that if we add both log-probability vectors together, p_L being mostly uniform and p_R having a clear maximum, we will get a disparity prediction that is informative. So, in the cases where finding the best match is easier in either N_L or N_R, having the two branches makes our model much more consistent in its predictions. This will also be shown in section 4.4.3 where we compare our model with the individual subnetworks and show that there can be variation in performance between N_L and N_R.

4.3.2 Siamese Network

This module is the same for both subnetworks N_L and N_R. As stated in section 4.3.1, the only difference is the domains of the input patches, being either RGB-LWIR or LWIR-RGB. In this section, we will use the left branch (N_L) to showcase the inner workings of the CNN module which is shown in figure 4.2. This module is a siamese network that takes a pair of stereo images as inputs $P_{N_L}^{RGB}$ and $P_{N_L}^{LWIR}$ and is inspired by [8]. Each patch is forwarded into a network of six layers composed of convolutions, batch normalization [2] and ReLU [3]. There is no ReLU activation for the last layer. For the convolutional layers, we used 32 filters of size 7×7, for the first two layers, and 64 filters of the same size for the remaining four layers. We also applied Dropout [71] to reduce overfitting because the datasets that we were working with are somewhat small.

After the left patch $P_{N_L}^{RGB}$ exits the sixth and last layer, we are left with a 64 dimensional feature vector $\vec{v}$. For the right patch $P_{N_L}^{LWIR}$, we have a feature volume of size $p_r \times 64$, named $\boldsymbol{M}$. We then do a correlation operation to obtain a vector of scores $\vec{s}$ of size p_r, with one score for each possible disparity location. We compute $\vec{s}$ as follows:

$$\vec{s} = \vec{v}\boldsymbol{M} \tag{4.4}$$

This means that each element s_i of $\vec{s}$ is the result of the dot-product between the feature vector $\vec{v}$ from the left patch $P_{N_L}^{RGB}$ and the column i of the feature volume $\boldsymbol{M}$ given by the right patch $P_{N_L}^{LWIR}$. Once we have our vector of scores, we feed it into a *LogSoftMax* layer to obtain the log-probability vector $\log(p_L)$ that is the output of one the CNN module.

As stated before, the same process is done in the network N_R with the left input $P_{N_R}^{LWIR}$ and the right input $P_{N_R}^{RGB}$ to get $\log(p_R)$.

4.3.3 Training

During training, we extract the small patches $P_{N_L}^{RGB}$ and $P_{N_R}^{LWIR}$ at pixel locations (x, y) for which we know the true disparity d. We also take the larger patches as $P_{N_L}^{LWIR}$ and $P_{N_R}^{RGB}$ at pixel locations $(x + d, y)$. These pair of patches are then forwarded into our network as presented in section 4.3.2. We use the Adagrad [72] optimizer during backpropagation.

For our loss function, we minimize the cross-entropy with regards to the weights w just as [8] did. Our loss function is defined as follows:

$$loss = \min_{\boldsymbol{w}} \sum\nolimits_{i,y_i} p_{gt}(y_i) \log p_i(y_i, \boldsymbol{w}) \tag{4.5}$$

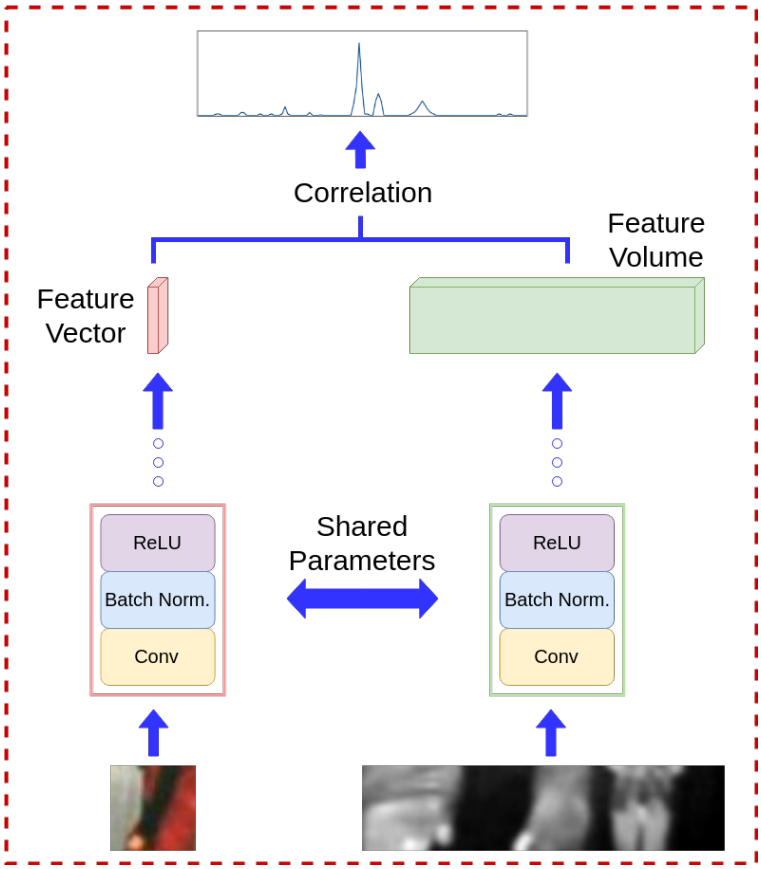
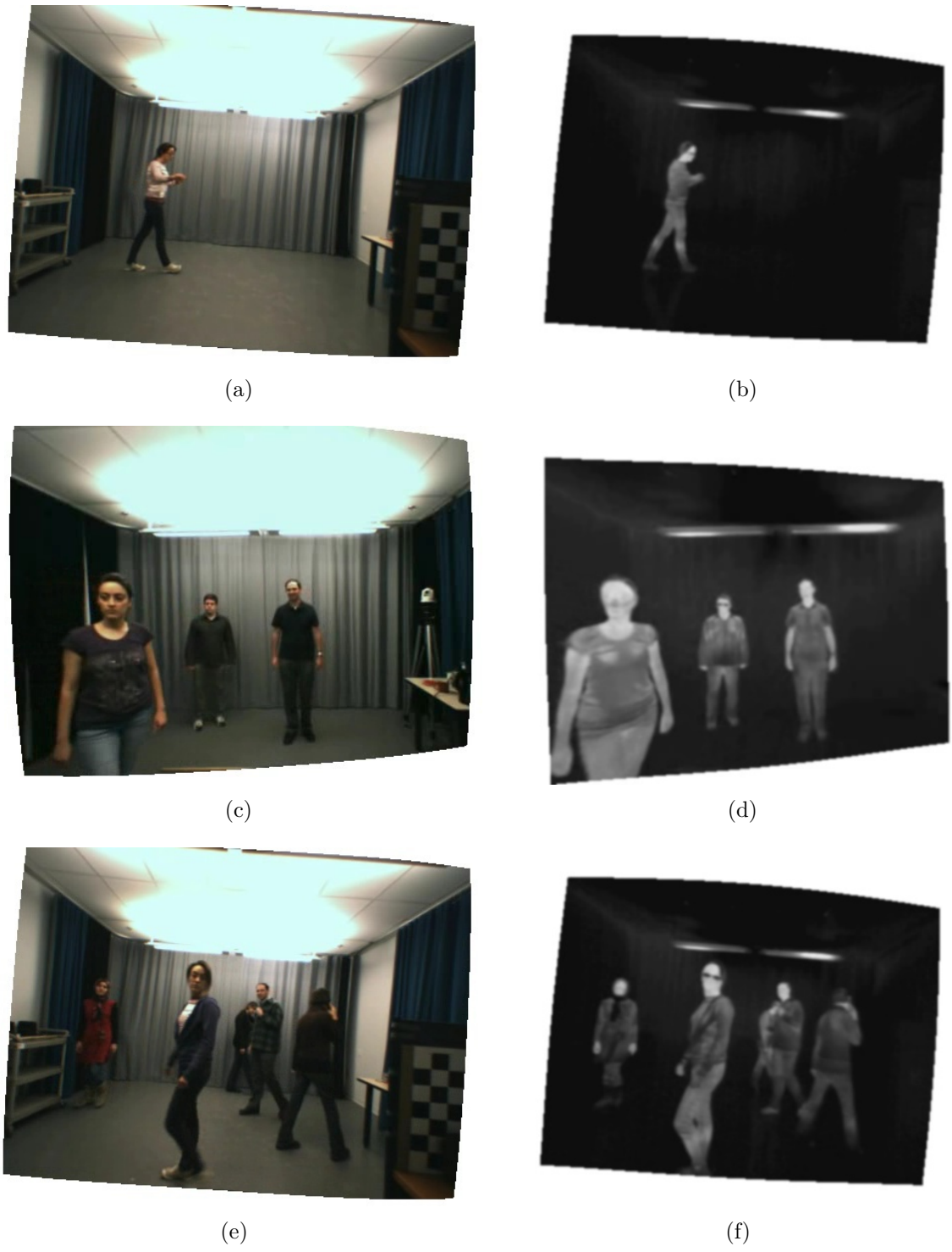


Figure 4.2 Details of the siamese network (N_L)

We also smooth the target distribution centered around the value of the ground-truth disparity. Since we are interested in the 3-pixel error metric, the possible values for the target distribution are defined as:

$$p_{gt}(y_i) = \begin{cases} \lambda_1, & \text{if } |y_i - y_i^{GT}| = 0 \\ \lambda_2, & \text{if } |y_i - y_i^{GT}| = 1 \\ \lambda_3, & \text{if } |y_i - y_i^{GT}| = 2 \\ \lambda_4, & \text{if } |y_i - y_i^{GT}| = 3 \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

In this work, we set these values as $\lambda_1 = 0.32$, $\lambda_2 = 0.40$, $\lambda_3 = 0.20$ and $\lambda_4 = 0.08$. With this target distribution, we can be sure that the network learns to minimize the disparity prediction error.



(a) (b)

(c) (d)

(e) (f)

Figure 4.3 Some examples of the multispectral stereo image pairs found in the LITIV dataset, one row being a different video sequence. First column is in the RGB domain while the second column is in the LWIR domain.

(a) (b)

(c) (d)

(e) (f)

Figure 4.4 Example of a multispectral stereo image pair found in the St-Charles *et al.* dataset. Each row corresponds to a different video sequence featuring from one to three subjects in each of them.

4.4 Experiments

This section will present the datasets used to conduct our experiments and present the various hyperparameters used for our model. We will also discuss the different manipulations that we did on our data in order to augment the original datasets. Finally, there will be a discussion about the obtained results.

4.4.1 Experiment Details

During training, we used two different datasets, one being the LITIV dataset [1] and the other one being the St-Charles *et al.* [4]. All images, in both datasets, were rectified so the search for the matching point will be in one dimension *i.e.* at the same y coordinate in the images of the stereo pair. The LITIV dataset, showed in figure 4.3 contains three video sequences, named vid01, vid02 and vid03, each of them consisting of people walking in a room at different depths. The number of person in a scene varies between one and five. The main difficulty in these videos are the occlusions between the different subjects and the visual appearance between the modalities. The video sequences respectively contain 89, 67 and 53 annotated images. Disparities are annotated for 11 166 points in vid01, 7529 points in vid02 and 6524 points in vid03. The St-Charles *et al.* dataset, showcased in figure 4.4 is also separated in three video sequences (v04, v07 and v08) and consists of one to three subjects walking in a room at different depths. Once again, the main difficulty is the amount of occlusion between the people walking in the room and the visual appearance between the modalities. The first sequence, v04, contains 117 images and 4252 disparity points, the second sequence, v07, has 144 images and 5653 disparity points and the last sequence, v08, contains 89 images for 5277 disparity points.

Table 4.1 Number of points used for training, evaluating and testing for both the LITIV and St-Charles *et al.* dataset. These number of points are after the data augmentation of section 4.4.2. The three different folds result in the testing of our method over all the images in the dataset.

	LITIV dataset			St-Charles dataset
	Training	Evaluation	Testing	Training
Fold 1	96 516 (vid01, vid02)	10 000 (vid01, vid02)	32 106 (vid03)	65 310 (vid04, vid07, vid08)
Fold 2	67 608 (vid02, vid03)	10 000 (vid02, vid03)	60 978 (vid01)	65 310 (vid04, vid07, vid08)
Fold 3	80 622 (vid01, vid03)	10 000 (vid01, vid03)	45 678 (vid02)	65 310 (vid04, vid07, vid08)

Our model was compared against handcrafted feature descriptors that required no training and that were tested on the LITIV dataset [1], so we had to separate the dataset in three

different folds in order to do a fair comparison with the other handcrafted feature methods and we applied a three-fold cross-validation. Table 4.1 summarizes the separation for each fold. The St-Charles *et al.* dataset was always part of the training data exclusively, while the LITIV dataset was separated between training, validation and testing. One important thing to notice is that the amount of training and testing points differ greatly depending on which video sequence we use, while the number of validation points always stay the same. We had to use exactly 10 000 validation points because we were limited by the memory of our GPU. The fold 1 is trained on all the St-Charles *et al.* data plus the images from vid01 and vid02, minus 10 000 randomly selected images kept for the validation. Vid03 was kept for the testing. For fold 2, the training set was composed of all the image pairs of the St-Charles *et al.* dataset with the addition of vid02 and vid03, once again, minus 10 000 randomly chose image pairs kept for the validation set. We tested on all the images in vid01. For the last fold, the training data was taken from the entire St-Charles *et al.* dataset in combination of vid01 and vid03 of the LITIV dataset, excluding the 10 000 image pairs that were put aside for the validation. The testing was made on the video sequence vid02 of the LITIV dataset. With these three folds, we are able to test our model on all three video sequences vid01, vid02, vid03. With a weighted average, we will be able to compare our method with the handcrafted feature descriptors that were all tested on the entirety of the LITIV dataset.

For all three folds, we kept the same hyperparameters. We will list the ones that are the most important *i.e.* the ones that have an impact on the results. The first ones are p_{hs} and p_{hr}, corresponding, respectively, to half the size and half the range of one patch. They are set to 18 and 60 respectively. As stated in section 4.3.1, these are the two parameters that control the size of the inputs of the CNN module. Dropout was also applied on our model. We used a drop-rate of 0.5 for all six layers, except the input layer which has a drop-rate of 0.2 as was suggested in the paper [71]. The learning rate α was set to 0.001 and was decreased after 24 000 iterations by a factor of $\frac{\alpha}{5}$. After that, every 4000 iterations, the learning was further decreased by the same rate. We trained our network for a total of 40 000 iterations, which correspond to 20 epochs. The batch size was set to 64.

4.4.2 Data Augmentation

One of the difficulties of working with stereo image pairs of both visible and infrared domains is that it can be quite hard to come by large datasets. We worked with both the LITIV dataset and the St-Charles *et al.* dataset and together, they contained a little more than 40 000 points. However, since we worked with patches, some of these points that were close to the images border were not valid, so in reality, we had less points to train our model. In

Table 4.2 Ablation study results of our model with three network configurations: with both subnetworks N_L and N_R, and with each of them individually. **Boldface**: best results.

	Recall		
	Proposed model	Left subnetwork (N_L)	Right subnetwork (N_R)
Fold 1	**0.839**	0.753	0.705
Fold 2	**0.783**	0.663	0.642
Fold 3	**0.731**	0.555	0.614

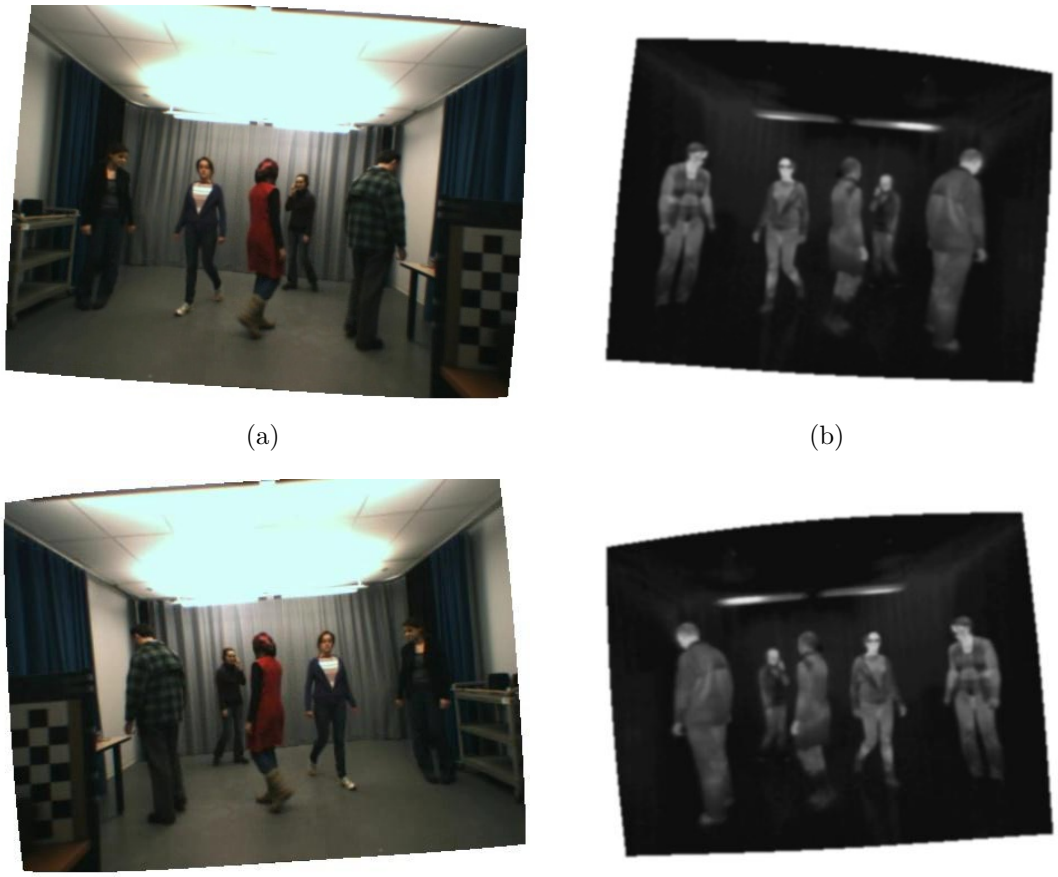
order to have more data to work with, we did some data augmentation operations. The first manipulation we did was to mirror every image in both dataset, which simply doubles the amount of data available. An example of a mirrored RGB-LWIR stereo image pair is shown in figure 4.5. The other operation applied on the data is that for each ground-truth point $p = (x, y)$ which we know the disparity d, we created two neighboring points one pixel above and one pixel below. Formally, we created the point $p_a = (x, y + 1)$ and $p_b = (x, y - 1)$ with the same disparity d (we are assuming that neighboring points should most of the time have the same disparity). This has the effect to triple every point, so with these two data augmentation operations, we have multiplied by a factor of six the amount of data that we can work with. Concretely, we now have a total of 136 300 points for the LITIV dataset and 65 310 points for the St-Charles dataset. Although we could benefit of having more diversified data, the quantity that we now have is sufficient to train and test our model effectively.

4.4.3 Results

We used the recall metric to do the comparison between our model and the handcrafted feature descriptors. In this case, the recall is defined as the correct number of matches on

Table 4.3 Results of our model, average over the three folds, compared with best handcrafted feature descriptors as reported in [1]. Patch sizes are in parentheses. **Boldface**: best result, *italic*: second best.

Method	Recall
Proposed model (37×37)	*0.779*
Mutual Information (40×130)	**0.833**
Mutual Information (20×130)	0.775
Mutual Information (10×130)	0.649
Fast Retina Keypoint (40×130)	0.641
Local Self-Similarity (40×130)	0.734
Sum of Squared Differences (40×130)	0.656



(a) (b)

(c) (d)

Figure 4.5 Example of the mirror operation of the data augmentation. The first row shows the original multispectral stereo image pair while the second row shows the mirrored image pair.

the total number of matches. Formally, using the 3-pixel error, we can define the recall as:

$$recall = \frac{m_{correct}}{m_{total}} \tag{4.7}$$

where $m_{correct}$ is the number of correct matches and m_{total} is the total number of possible matches. A match is considered correct if the predicted disparity y is at exactly three pixels or less than the ground-truth disparity.

Table 4.2 shows the results of our method compared with the performance of each individual subnetwork, where the prediction of either N_L or N_R is considered as the final disparity prediction y. We evaluated these three networks on all three folds that were presented in table 4.1. As we can see, this ablation study shows that the results of our proposed model are always better than the individual subnetworks. For the first fold, our model with the two

subnetworks achieves a better recall by a margin of 8% when compared to the best individual subnetwork. The margin is even bigger for the second and third folds, being of 12% and 11% respectively. Therefore, it demonstrates that the use of two parallel subnetworks help improve the performance of our model. This is expected because, as we explained earlier in section 4.3.1, when working with the two branches, our model can learn to have a similar prediction as the output of both subnetworks N_L and N_R which gives us a high confidence in the disparity prediction by verifying the left-right consistency. When working with only one branch (either N_L or N_R), whatever the prediction, we have to take it as final, which will lead to more errors than the case where we have the predictions of both branches.

Table 4.3 compares the handcrafted feature descriptors as reported in [1] with our proposed method. We took as a comparison, the best results reported. Therefore, the patches for the handcrafted features include more pixels as the patches have the same width as us, but are about three times higher. The recall for our network is a weighted average with the weights being proportional with the number of evaluated points from table 4.1. This way, we can have a fair comparison between our model and the descriptors, which were evaluated on all three videos of the LITIV dataset. Globally, we see that our model ranks in second place across all methods, being surpassed only by mutual information (40×130). We believe this happens for a number of reasons. First of all, we reported the best results for each method, which were obtained with window sizes of 40 x 130. This window size is similar to a the size of a human silhouette in an image. Thus, these methods had more relevant information to work with than our method with patches of size $p_s \times p_s$ where $p_s = 37$. We believe this can lead to better recall results for mutual information [29]. In fact, with patches containing the same number of pixels as ours, about 1300, the results of mutual information are lower than ours with a recall of 0.649 for 10×130 patches. With twice the number of pixels as ours, that is 20×130 patches, our method is marginally better with a recall of 0.779 when compared to 0.775 of mutual information.

Second, if we look at table 4.1, we see that there are variations in the results of our method depending on the folds. Between the best fold and the worst one, there is a difference of almost 11%, with the other fold being in the middle of the two with a difference of close to 6% for the best fold and 5% with the worst one. If we refer to table 4.1 with all the folds data separation, we can see that the training set sizes vary between all folds. Although the difficulty of the test samples in the folds may vary, we remark that the best recall results were obtained on the fold with the most training samples available while worst results were obtained with less training data. This leads us to believe that with more data available, we would be able to get better results because, with the current folds, there is a correlation with the recall results and the amount of training data. This is why, in the case of the first fold,

we are able to get slightly better result than mutual information, but that is not the case for the second and third folds.

Even with achieving second place across all methods, we believe that our model is competitive with the handcrafted feature descriptors and with more data available, we could get event better performance. We believe this consists evidence that CNNs are able to, in the context of disparity estimation between visible and infrared domains, be competitive with the current feature descriptors.

4.5 Conclusions

In this paper, we presented a new model capable of doing disparity estimation between pairs of visible-infrared images. Our model is made of two subnetworks, each being a siamese network and outputting a log-probability vector for each possible disparity location. These two log-probability vectors are then summed together to get the final prediction. The goal of having two subnetworks working in parallel is to enforce left-right consistency in order to be more confident the final disparity prediction. We used two datasets, one used exclusively for training while evaluating on the other. Because of a lack of data, we did two data augmentation operations and performed a three-fold cross-validation. We demonstrated that we can achieve competitive performance with our proposed model when compared to handcrafted feature descriptors.

Acknowledgements

This work was supported by a scholarship from the National Sciences and Engineering Research Council of Canada (NSERC) and a grant from the Fonds de recherche du Quebec - Nature and Technologies (FRQNT). We thank NVIDIA Corporation for their donation of a Titan X GPU used for this research. Finally, a special thank you to Pierre-Luc St-Charles for his help rectifying the stereo images of his dataset.

CHAPITRE 5 ARTICLE 2 : DOMAIN SIAMESE CNNs FOR SPARSE MULTISPECTRAL DISPARITY ESTIMATION

D.-A. Beaupre and G.-A. Bilodeau,
publié à The IEEE International Conference on Pattern Recognition (ICPR), January 2021

Abstract

*Multispectral disparity estimation is a difficult task for many reasons: it has all the same challenges as traditional visible-visible disparity estimation (occlusions, repetitive patterns, textureless surfaces), in addition of having very few common visual information between images (*e.g. *colour information vs. thermal information). In this paper, we propose a new CNN architecture able to do disparity estimation between images from different spectra, namely thermal and visible in our case. Our proposed model takes two patches as input and proceeds to do domain feature extraction for each of them. Features from both domains are then merged with two fusion operations, namely correlation and concatenation. These merged vectors are then forwarded to their respective classification heads, which are responsible for classifying the inputs as being same or not. Using two merging operations gives more robustness to our feature extraction process, which leads to more precise disparity estimation. Our method was tested using the publicly available LITIV 2014 and LITIV 2018 datasets, and showed best results when compared to other state-of-the-art methods.*

5.1 Introduction

Disparity estimation from stereo images is one of the fundamental tasks in the field of computer vision. It can be used to predict the depth in a scene, register images into the same coordinate system and perform object detection. It also has many real-world applications such as for autonomous vehicles and for 3D model reconstruction [73, 74].

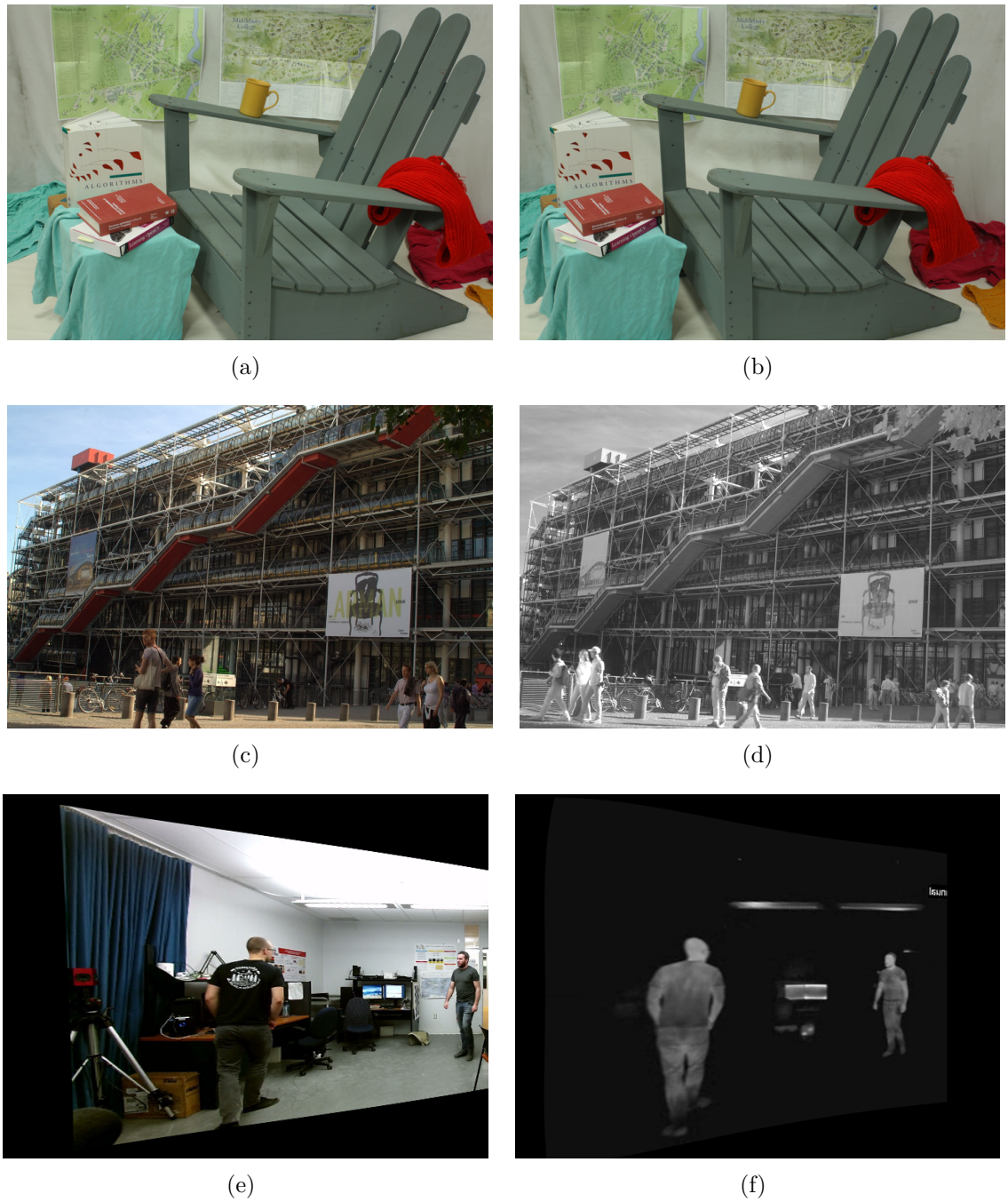
Many works in the literature focus on disparity estimation in the visible (RGB) domain [8,9,41,45,67], where both images are captured with RGB cameras. Recently, more works use multispectral image pairs, where one of the images is in the infrared (IR) spectrum [11,56,75]. This has both advantages and disadvantages. For instance, when we want to detect a given object that has a temperature different from the environment, working with thermal IR images can be beneficial since the desired object will be easily detected. For example, humans

usually have a body temperature that is higher or lower than the ambient temperature in public places, making human detection easier. This is especially true if a person's clothes have a low contrast with the environment. For instance, at night time, if someone is walking in a public park, detecting him or her with a thermal IR camera will be easy. However, detecting the same person with a RGB camera would be a lot more challenging. The opposite is also true, where detecting a human in daylight will be easier than in thermal if this human has a body temperature similar to the environment (low thermal contrast). RGB provides as well more information to help describe and identify persons. Thus, if we work with two cameras from different spectrums, we can get the best of both worlds, since RGB cameras provide more visual information in the case of people, but they require appropriate lighting to do so, while IR camera can capitalize on the thermal contrast between humans and their environment, but will by default remove a lot of details like textures and colours from the image making people identification harder.

Many works in disparity estimation are designed to work for RGB cameras, where both images have similar content (colours, textures). Lately, several works focus on multispectral images where one image is in RGB while the other one is in the infrared domain. In this case, the amount of similar content in both images will depend on the type of infrared used. However, in all cases, working with IR images reduces the similarity between the two images of the stereo pair, which means that matching the content of said images is harder, and therefore disparity estimation is also harder.

In figure 5.1, we show three different image pairs. It showcases how similarities between image pairs diminish depending on the spectrum of the images. In the case of RGB-RGB images, we can notice that there are a lot of similarities between the two images. Generally, in this domain, the main difficulties are with repetitive patterns, occlusions and textureless areas. However, it is fairly easy to match pixels between two RGB images, as the results on public benchmarks are very good [59, 76]. Next, for the case of RGB-NIR (near infrared) images, we observe that there is less common information between the images, but it is still possible to note common objects between both spectrum, as textures are shared and object edges are well defined in both images.

Lastly, if we take a look at RGB-LWIR (long wavelength infrared or thermal IR) images, we see that there are very few common features between images, mainly the people whom we can match, but every object in the background, if not emitting heat, is very hard to see in the LWIR image. This makes the matching of pixels between both domains much harder than in the previous domain pairs stated above. Our proposed method operates on RGB-LWIR image pairs, which means that we need a method that is able to learn to match features



(a) (b)

(c) (d)

(e) (f)

Figure 5.1 First row: images from the Middlebury 2014 [13] stereo dataset where we can see a lot of common information between the two images. Second row: images from the VIS-NIR [5] dataset, which still shows similar textures between both images. Third row: images from the LITIV 2014 [1] stereo dataset where the only common information between the two images are the objects emitting heat, with very few common textures.

between those domains.

In this paper, we present a new convolutional neural network (CNN) architecture inspired by [45, 77] able to do disparity estimation between RGB-LWIR image pairs. Our model is a domain siamese network, meaning that each image of the stereo pair has its own feature extractor, but both feature extractors have the same structure *i.e.* there is no weight sharing between the branches of the siamese network. We are not able to do dense disparity estimation because of the nature of our datasets, which are not densely annotated. That is why we will work with patches instead of images. Our model takes two small square patches as input, and extracts features from those resulting in a feature vector for each image patch. We do two operations on the feature vectors: 1) we do a correlation product between both vectors and forward the result to the correlation head, 2) we do a concatenation between the two vectors and forward it to the concatenation head. Using two different complementary techniques gives more robustness to our network, leading to better performance when compared to using only correlation or concatenation. Both the correlation and concatenation heads consist of fully connected layers outputting the probability of both patches being the same or not. Each classification head has its own loss function, and at testing, we use both classification heads to get the disparity predictions. The source code of our method will be available online at `https://github.com/beaupreda` upon publication.

Our paper has the following two main contributions:

— We propose a new siamese CNN architecture able to do sparse disparity estimation between multispectral RGB-LWIR image pairs. Our architecture extracts features from both image domains and uses both concatenation and correlation to get a probability representing if the input patches are the same or not.
— Experiments performed on the LITIV 2014 [1] and LITIV 2018 [4] datasets show that our model is able to achieve better performance than all past methods, either using classical descriptors or methods based on CNNs.

5.2 Related Works

5.2.1 Multispectral Images

Until recently, the best way to do disparity estimation between the RGB and LWIR domains was with handcrafted feature descriptors, such as SIFT [48] or mutual information [29]. Mutual information is an example of a window-based method, meaning that the descriptor is computed as a similarity between pixels inside two candidate windows. The method consists in computing statistics based on the co-occurrences of the pixels inside the windows. This

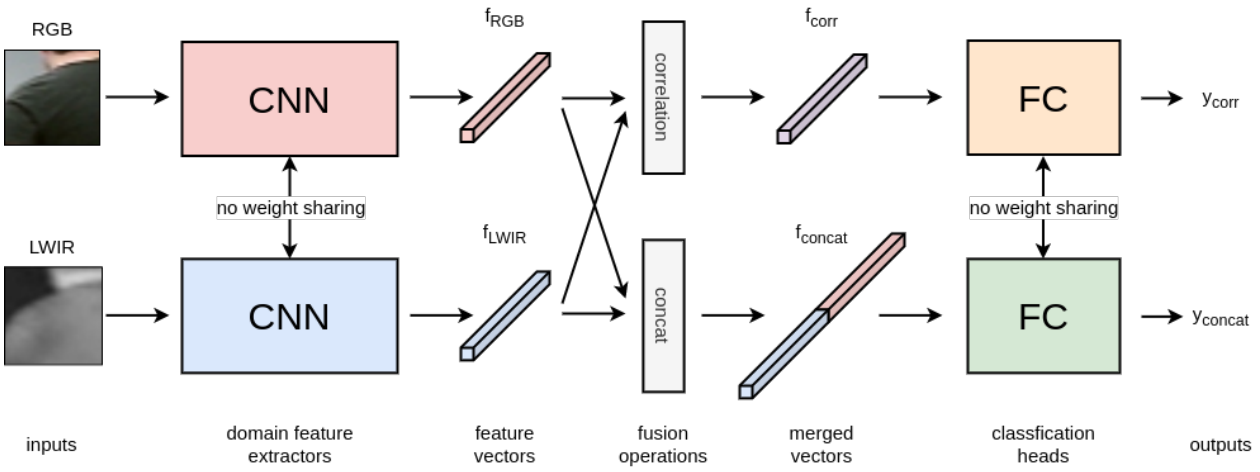


Figure 5.2 Details of the proposed architecture. We have two CNNs doing domain feature extraction to obtain feature vectors. These vectors are then merged with a correlation and concatenation operation before being passed to the classification heads, which are responsible for classifying the two inputs either as the same or not with a probability score.

method was able to get the best results among all handcrafted feature descriptors according to a study [1]. The sum of squared difference (SSD) [1] is another example of window-based method. Other methods like local self-similarity (LSS) [49], histogram of oriented gradients (HOG) [31] and SIFT [48] are based on modelling the distribution of data. LSS is a local descriptor that captures the self-similarity of regions with colours and textures, which makes it the best descriptor in its category [1]. SIFT and HOG are both based on gradients and are invariant to illumination changes which is beneficial when working with multispectral images.

In the last years, there were also works based on CNNs that were designed for patch matching between RGB and IR domains. It is to note that for visible/IR stereo pair datasets, there is not enough data to train CNNs for end-to-end dense stereo matching. The work of [56] gives an interesting comparison of different CNN architectures to do patch matching between the RGB and NIR domains. Aguilera *et al.* [11] propose a quadruplet network inspired by the popular triplet networks [78]. Quadruplet networks take four input patches, creating two positive pairs and four negative ones, leading to top performance on the VIS-NIR benchmark [5]. In the same task, AFD-Net [75] uses a model based on metric learning using the difference of features between the RGB and NIR images. Zhi *et al.* [12] worked with a new RGB-NIR dataset made up of road scenes. It is an unsupervised method that transforms the RGB image into a pseudo-NIR image and uses projection of the pseudo-image to do self-supervision. It

also uses segmentation to differentiate the materials in the scene. Beaupre *et al.* [77] proposed a dual siamese network, effectively working with four inputs to enforce consistency between the predictions of the left and right subnetworks.

5.2.2 Stereo Matching

We will focus on works that use CNNs since they are the most relevant to our work. Zbontar *et al.* [67] were the first to propose using a CNN to extract features from images instead of using handcrafted feature descriptors. They created a cost volume with the features from the CNN with SGM [79] to obtain the disparity map. Luo *et al.* [8] used a correlation product to merge features from the left and right images, resulting in much faster stereo matching than [67]. Kendall *et al.* [9] were the first to propose an end-to-end architecture for stereo matching which is now the default architecture choice for state-of-the-art methods on public benchmarks [59]. They used 3D convolutions to learn from the context of features, before doing a regression in order to get the disparity map. Chang *et al.* [41] also wanted to improve disparity estimation with help from the context, so they used a spatial pyramid pooling module to extract features at multiple scales and used an hourglass network to regularize the cost volume. The methods of [47] and [80] both train a network jointly with another task to improve disparity, namely segmentation and edge detection, respectively. Both papers hypothesize that most errors in disparity estimation come at object borders, so using another task that defines the boundary between objects will improve stereo matching. Zhang *et al.* [44] focus on the aggregation step by proposing two new layers: semi-global guided aggregation layer (SGA) and local guided aggregation layer (LGA). SGA is basically a version of SGM, but with learnable parameters, while LGA learns to refine thin structures. Guo *et al.* [45] focus on the cost volume step by creating two cost volumes: one from the concatenation of features and the other one from the correlation of features. These are then merged in a group-wise manner to form a combined volume of rich features, which leads to better performance than only using concatenation or correlation.

5.3 Method

This section presents an overview of the proposed method, consisting of a domain siamese network with a classification branch and a correlation branch, as well as an explanation of our training and testing methodology, and the implementation details. Figure 5.2 illustrates the global architecture of our model.

5.3.1 Network Architecture

Our network architecture is inspired by previous work in RGB-RGB disparity estimation, as well as RGB-LWIR disparity estimation. It is based on the popular siamese architecture in stereo estimation, where a network takes two inputs and extracts features from these. We modified this base architecture for our needs. Usually, siamese networks share the same feature extractor between the two inputs. However, in our case, we found that having a different feature extractor for each domain gave better results since RBG and LWIR images have different visual appearance. This same conclusion was reached in [52] for matching keypoints between the RGB and NIR domains.

Our architecture takes two square patches as inputs, one from the RGB image and the other from the LWIR image, which will be referred to as P_{RGB} and P_{LWIR} throughout the paper. Each patch is forwarded into their respective domain feature extractors, one for the RGB features, and the other one for the LWIR features. Both of these CNNs have the exact same structure, which is detailed in table 5.1, but each has their own weights. We know from numerous papers that weight sharing between siamese networks is a common practice, specifically in RGB-RGB stereo vision. However, we found that the model performed better when each patch had its own CNN to extract features. We suppose this is because, as mentioned in section 5.1, there is not a lot of common information between images of the RGB-LWIR domains. In the case of images in the RGB domain, sharing weights is logical since we want the network to learn to find similar features from the images. In our multispectral case, since the content of both images can be quite different, we believe it is preferable to extract features separately and then combine them with a fusion operation. The CNNs each produce a 256D feature vector, f_{RGB} and f_{LWIR}, representing P_{RGB} and P_{LWIR} respectively. With these feature vectors, we proceed to do two different fusion operations: correlation and concatenation. These two operations are the most common operations in disparity estimation to join features from both images. Each has advantages over the other. Correlation is usually faster to compute and consumes less memory than concatenation, but it loses information *e.g.* it does not keep all the feature information across the channel dimension. On the other hand, concatenation does not lose any information as the whole channel dimension is kept, at the cost of more memory consumption and longer computation time. It was shown that using both a correlation cost volume and a concatenation cost volume improves performance when compared to using only one of the two [45]. We take inspiration from this work to use both operations to form the merged feature vectors

$$f_{corr} = f_{RGB} \odot f_{LWIR} \tag{5.1}$$

Table 5.1 Details of our proposed architecture, layer by layer. Layer structure is under the form $k \times k, c$, where k represents the convolutional kernel size and c the number of channels. Output dimension is under the form $h \times w \times c$, h being the height of the patch, w its width and c, the number of feature channels. Every convolutional layer is followed by batch normalization [2] and has a ReLU [3] activation function, except for *conv9*. *fc1* and *fc2* also use the ReLU activation, while *fc3* uses a Softmax activation to get probabilities.

Name	Layer structure	Output dimension
input		$36 \times 36 \times 3$
CNN		
conv1	5×5, 32	$32 \times 32 \times 32$
conv2	5×5, 64	$28 \times 28 \times 64$
conv3	5×5, 64	$24 \times 24 \times 64$
conv4	5×5, 64	$20 \times 20 \times 64$
conv5	5×5, 128	$16 \times 16 \times 128$
conv6	5×5, 128	$12 \times 12 \times 128$
conv7	5×5, 256	$8 \times 8 \times 256$
conv8	5×5, 256	$4 \times 4 \times 256$
conv9	4×4, 256	$1 \times 1 \times 256$
FC		
fc1	$256/512, 128$	1×128
fc2	$128, 64$	1×64
fc3	$64, 2$	1×2

and

$$f_{concat} = [f_{RGB}, f_{LWIR}]. \tag{5.2}$$

f_{corr} is computed as the element-wise product between the feature vectors f_{RGB} and f_{LWIR}, so it remains a 256D vector. On the other hand, f_{concat} is a 512D vector, since both f_{RGB} and f_{LWIR} are concatenated. Both merged vectors are then forwarded into their respective classification head, one for correlation and the other for concatenation. Their structure is detailed in table 5.1 and both of them outputs a 2D probability vector, representing the probability that the two patches are the same or not. This is therefore a binary classification problem.

5.3.2 Training

The first step in training the network is to extract training patches P_{RGB} and P_{LWIR} around known ground-truth pixels in rectified stereo pairs. P_{RGB} will be centered on pixel p at

location (x, y), while P_{LWIR} will be centered on pixel q at location $(x + d, y)$, accounting for the disparity d, x and y representing the pixel coordinates in the image space. To make our network more robust, we consider as positive matches, two patches that are positioned from one another at $d \pm 1$ in x.

Since we are considering the task as a binary classification problem, we also need samples of negative pairs. In order to create negative pairs, we start by taking all the positive patches located at (x, y) and $(x+d, y)$ and add an offset o to the x values. The range of o is $[-30, -10]$ and $[10, 30]$, meaning that we consider negative samples either to the left or the right of the ground-truth pixels. For every positive pair, we create a negative pair at $(x + o, y)$ and $(x - d + o, y)$ so we have a balanced dataset with as many positive samples as negative ones. The value of o is determined randomly following a uniform distribution for each example. All positive and negative samples are then shuffled together.

We use two binary cross-entropy loss functions to train our network, one for the correlation branch, and the other one for the concatenation branch. They are given by

$$loss_{corr/concat} = -\frac{1}{N}\sum_{i=1}^{N} gt_i log(y_i), \tag{5.3}$$

where N represents the number of samples, gt_i the ground-truth value, being either 1 if the patches are the same or 0 if they are not, and y_i the probability of patches being the same or not. The total loss is then given by

$$loss = loss_{corr} + loss_{concat}. \tag{5.4}$$

5.3.3 Prediction

Since our goal is to predict a disparity, we need to add some operations on top of our network to obtain it. For prediction, we expand the width of P_{LWIR} to take into account the maximum disparity parameter $disp_{max}$. Half of $disp_{max}$ is added to the width on both

Table 5.2 Details of the LITIV 2014 data separation into three folds as well as the number of points with data augmentation and from which video these ground-truth points are taken.

	Training		Validation	Testing
	LITIV 2018	LITIV 2014	LITIV 2014	LITIV 2014
Fold 1	218 230 (vid04 + vid07 + vid08)	240 167 (vid02 + vid03)	35 378 (vid02 + vid03)	101 433 (vid01)
Fold 2	218 230 (vid04 + vid07 + vid08)	291 720 (vid01 + vid03)	34 688 (vid01 + vid03)	76 001 (vid02)
Fold 3	218 230 (vid04 + vid07 + vid08)	320 648 (vid01 + vid02)	34 220 (vid01 + vid02)	61 771 (vid03)

Table 5.3 Details of the LITIV 2018 data separation into three folds as well as the number of points with data augmentation and from which video these ground-truth points are taken.

	Training		Validation	Testing
	LITIV 2014	LITIV 2018	LITIV 2018	LITIV 2018
Fold 1	478 410 (vid01 + vid02 + vid03)	109 620 (vid07 + vid08)	44 226 (vid07 + vid08)	32 192 (vid04)
Fold 2	478 410 (vid01 + vid02 + vid03)	91 904 (vid04 + vid08)	49 286 (vid04 + vid08)	38 520 (vid07)
Fold 3	478 410 (vid01 + vid02 + vid03)	99 858 (vid04 + vid07)	41 566 (vid04 + vid07)	38 403 (vid08)

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.3 First two columns: images from the LITIV 2018 [4] dataset. Last two columns: images from the LITIV 2014 [1] dataset. These images showcase some of the difficulties present in both datasets. Image pairs (a)-(e) and (d)-(h) show an example of occlusion while pairs (b)-(f) and (c)-(g) show an example of textures not visible in the LWIR domain.

sides of the center of the original P_{LWIR} patch. P_{RGB} is kept the same size. We forward both patches into the CNNs, leaving us with f_{RGB} a 256D vector and f_{LWIR} a $256 \times disp_{max}$ tensor. Now, for each disparity d. we extract the corresponding feature vector from f_{LWIR} leaving us with two vectors, each of 256D. We then forward both vectors into the FCs layers to obtain probabilities of the vectors being the same or not. We do this process for every disparity d, and only keep the probability of both patches being the same, which leaves us with a probability vector p of size $disp_{max}$ of the patches being the same at every disparity. We do a disparity regression as in [9] *i.e.* a weighted sum of the normalized probabilities multiplied by d to obtain our disparity predictions $\hat{d}_{corr}$ and $\hat{d}_{concat}$ for each head, respectively with

$$\hat{d}_{corr/concat} = \sum_{d=0}^{disp_{max}} d \times p_d. \tag{5.5}$$

$\hat{d}$, the final disparity prediction, is the mean of the disparity predictions of both branches and is given by

$$\hat{d} = \frac{d_{corr} + d_{concat}}{2}. \tag{5.6}$$

5.3.4 Implementation details

We implemented our network with the PyTorch [81] framework, and as stated earlier, the code will be made publicly available online at `https://github.com/beaupreda`. The default patch size is 36×36 and the maximum disparity used for testing is 64. We use the Adam [82] optimizer to train our network with backpropagation and a starting learning rate α of 0.01 which is updated to $\frac{\alpha}{2}$ every 40 epochs. We trained our model on every fold of data for 200 epochs, which takes a little more than 12 hours for each fold on a single NVIDIA Titan X.

5.4 Experiments

This section will present in detail the datasets we used to train and test our model, as well as the data augmentation techniques we used to get more data. It will also show the results we got and discuss them with a comparison against other methods.

5.4.1 Datasets

We use two datasets to train our network and do the evaluation, namely the LITIV 2014 dataset [1] and the LITIV 2018 dataset [4]. Some examples of images found in both datasets are shown in figure 5.3. These images show some of the difficulties of the datasets, like occlusions, where the occluded person is less visible in one of the spectrum, making the matching of patches harder. Another difficulty is the difference of textures between the RGB and the LWIR domains, where a checkered shirt appears as of uniform appearance in the LWIR domain. Another difficulty comes from the number of persons in a given scene, since the number of potential matches increases with the number of people in a video.

The LITIV 2014 dataset is made of three videos, named vid01, vid02 and vid03, each presenting scenes with people with annotated disparities. The number of subjects in each video varies from one to five, and they are all walking at different depths in the scene, creating

occlusions, which is one of the difficulties in this dataset. The videos respectively have 89, 67 and 53 image pairs for a total of 11 166 points in the first video, 7529 points in the second and 6524 points in the third. The LITIV 2018 is also separated in three videos, named vid04, vid07 and vid08. These videos feature one to three subjects walking in a scene with the additional difficulty of them manipulating some objects emitting heat, such as a kettle. There are 117 images for 4252 ground-truth points in the first video (v04), 144 images for 5653 ground-truth points in the second video (v07) and 89 images for 5277 ground-truth points in the third sequence (v08).

We compared the performance of our model against classical methods that do not need any training phase, so these were tested on all available videos in the dataset. However, in our case, we need to separate data into three sets, namely training, validation and testing. Since we want to have a fair comparison against all methods, we split both datasets into three folds, each fold having a distinct testing set as to have the complete dataset tested. Tables 5.2 and 5.3 show the separations of the folds with the number of data points in each fold for the LITIV 2014 and LITIV 2018 datasets, respectively. The separation scheme is fairly simple: for a given dataset, we keep one video for testing and use the other two for training and validation. The other dataset is used as training data. For the validation sets, we selected randomly 30 images for LITIV 2014 and 150 images for LITIV 2018.

5.4.2 Data Augmentation

One problem with RGB-LWIR multispectral datasets is that there is not a lot of available data to train CNNs. For instance, in our case, we use both LITIV datasets, and obtain a little more than 40 000 ground-truth points without any data augmentation. However, with the way we train our network and two data augmentation operations, we are able to effectively increase by a factor of 20 the amount of training data.

The first data augmentation operation that we did is what we call the cross duplication. This process is illustrated in figure 5.4 and basically consists in giving the same disparity to neighbours of a given ground-truth pixel. If we have a ground-truth pixel p at location (x, y) with disparity d, we simply create four new ground-truth points with the same disparity d at locations $(x \pm 1, y \pm 1)$, giving the shape of a cross to our ground-truth points. We remove any duplicates that this process can create, as two ground-truth point having a common neighbour will duplicate a new point. This operation basically adds a factor of five to the number of original disparity points. One important thing to note, however, is that from the original 40 000 points, some of them cannot be used in this step since they are too close to the image border, and we cannot extract patches around those points. Since all our points

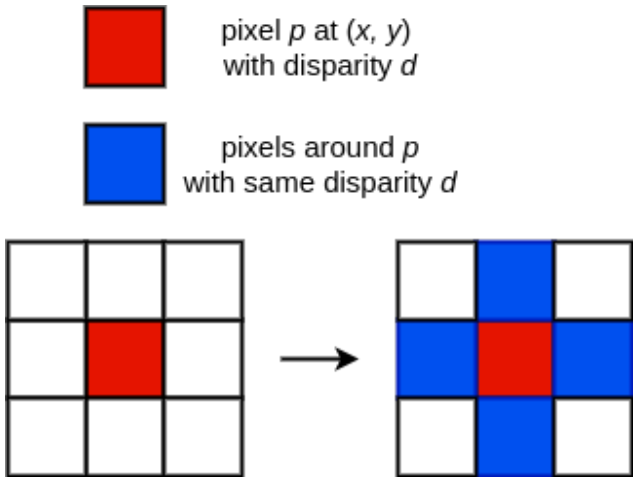


Figure 5.4 Illustration of our cross data augmentation method. Basically, we add four training points from one pixel from which we have the ground-truth, giving us more training data to reduce overfitting.

Table 5.4 Ablation study on the LITIV 2014 dataset showcasing the difference of performance between our proposed model (last columns) and using only one of the two fusion operations (first and second columns). **Boldface**: best results

	Correlation branch only			Concatenation branch only			Corr + Concat (proposed model)		
	≤ 1 px	≤ 3 px	≤ 5 px	≤ 1 px	≤ 3 px	≤ 5 px	≤ 1 px	≤ 3 px	≤ 5 px
Fold 1	0.524	0.859	0.984	0.551	0.894	0.981	**0.588**	**0.901**	**0.985**
Fold 2	0.454	0.854	0.978	0.472	0.897	0.985	**0.474**	**0.904**	**0.986**
Fold 3	0.541	0.875	0.982	0.558	0.895	0.982	**0.629**	**0.916**	**0.989**

are on people, this does not happen very often, mostly when a subject enters the scene.

The second data augmentation operation we performed is the mirroring of images *i.e.* flipping the images around the y axis. This operation doubles the amount of data points from the vanilla data. Now, if we combine both data augmentation techniques and use the fact that we have to create negative samples for each ground-truth point that we have (doubles the number of data), these operations give us a factor of 20 when compared to the original training data. With this amount of training data, we reduce the chance of overfitting and increase the robustness of our network.

5.4.3 Results

The performance measure we use to compare our model to other methods is the recall, which computes the number of predictions $\hat{d}$ that our network made that is at a distance of t pixels

Table 5.5 Comparison of our model against two other methods on the LITIV 2018 dataset. **Boldface**: best result.

	Fold 1		Fold 2		Fold 3		Overall	
	≤ 1 px	≤ 4 px	≤ 1 px	≤ 4 px	≤ 1 px	≤ 4 px	≤ 1 px	≤ 4 px
DASC Sliding Window [4]	0.104	0.265	0.086	0.236	0.121	0.289	0.104	0.263
Multispectral Cosegmentation [4]	0.253	0.562	0.236	0.531	0.307	0.678	0.265	0.590
Proposed Model	**0.480**	**0.943**	**0.446**	**0.877**	**0.406**	**0.972**	**0.442**	**0.930**

Table 5.6 Comparison of our model against other methods on the LITIV 2014 dataset. Patch sizes are in parentheses. **Boldface**: best result, *italic*: second best.

Method	≤ 3 px
Proposed Model (36×36)	**0.906**
Siamese CNNs [77] (37×37)	0.779
Mutual Information [1] (40×130)	*0.833*
Mutual Information [1] (20×130)	0.775
Mutual Information [1] (10×130)	0.649
Fast Retina Keypoint [1] (40×130)	0.641
Local Self-Similarity [1] (40×130)	0.734
Sum of Squared Differences [1] (40×130)	0.656

or less from the ground-truth. We formally define it by

$$recall = \frac{1}{N} \sum_{i=1}^{N} |\hat{d}_i - gt_i| \leq t, \tag{5.7}$$

where N stands for the number of samples, $\hat{d}_i$, the disparity prediction for the ith example and gt_i the ground-truth of the ith example. This evaluation methodology is the same as the other methods that we compare ourselves to, so we can directly report the results from those papers.

Table 5.4 presents the results from our ablation study which evaluates the performance of our model when only one fusion operation is considered (either only correlation or only concatenation). For the correlation branch only results, we trained our network without the concatenation branch. The same process was applied to the concatenation only results, where only the concatenation branch was considered during training. The last column is our proposed model with both branches present during training, and the final disparity prediction $\hat{d}$ is the average of the predictions of both heads. The most important result from this table is that using both branches leads to better results for disparity prediction. This is expected since using both correlation and concatenation leads to better extracted features by the CNNs, and therefore better predictions. We also observe that by itself, the

concatenation branch has better results than the correlation branch. This result is logical since the concatenation operation keeps more features (512D feature vector) compared to the correlation operation (256 feature vector). Also, it is the same conclusion reached in RGB-RGB disparity estimation where networks who build a concatenation cost volume have better performance than the ones who use a correlation operation.

Table 5.6 shows the results of our method compared to a CNN-based method from [77] and several methods based on handcrafted feature descriptors, as reported in [1]. The recall obtained by our proposed model is the weighted average (by the number of test points) of the recalls obtained for the three separate folds reported in table 5.4. We can see that our proposed method surpasses every other method by a large margin. We improve the past results based on deep learning by a little bit more than 0.12, while also having an improvement of around 0.07 over mutual information [29], which was the best on the LITIV 2014 dataset. We also achieve this performance while working with fewer pixels than any other methods. Our performance is significantly superior to siamese CNNs with around the same number of pixels to make our decision, but if we compare our small 36×36 patches to the other methods, we can notice that they use up to four times more pixels, and yet, we are able to obtain better disparity predictions.

Table 5.5 presents the results on the LITIV 2018 dataset against two approaches, one being classical, while the other is a sophisticated cosegmentation method applying belief propagation to optimize disparity estimation and segmentation jointly. Here, we can see that our method is superior to the other two approaches, having an overall better performance of more than 0.20 for the smallest threshold, and more than 0.30 for the bigger one.

We believe this demonstrates that our method is robust and that CNNs-based methods are able to be competitive on hard tasks like disparity estimation between RGB-LWIR image pairs. We believe that these performances are due to mainly two features of our model: first, the domain feature extractors, which are responsible to get features from both domains separately and the use of two fusion operations which forces the network to learn better representations of the multispectral images.

5.5 Conclusion

In this paper, we presented a new model able to do sparse multispectral disparity estimation between image pairs from the RGB and LWIR domains. Our model uses a siamese domain feature extractor, which extracts features independently for both images, as opposed to traditional siamese networks who share the same weights for both images. We believe that it is

preferable to keep the feature extractions separate in the case of images from different spectral domains. We use two operations to merge the features extracted by our CNNs, namely correlation and concatenation. Using the two jointly is shown to improve the performance of our network. We believe that by using the two fusion operations, we augment the learning capability of our network and make it more robust which leads to better performance. Evaluation on public datasets shows that our method is significantly better than all the other methods tested for sparse multispectral disparity estimation.

Acknowledgment

This work was made possible by a scholarship from the National Sciences and Engineering Research Council of Canada (NSERC). We would also like to thank NVIDIA Corporation for their donation of a Titan X GPU used for this research.

CHAPITRE 6 DISCUSSION GÉNÉRALE

Ce chapitre effectuera une comparaison des deux méthodes développées dans ce mémoire, présentera certains problèmes de nos méthodes, et discutera de certaines architectures essayées, mais qui ne donnaient pas les résultats escomptés.

Une première observation par rapport aux architectures développées est que la première est faite de deux sous-réseaux siamois, tandis que la deuxième possède qu'un seul réseau siamois. Dans la première méthode, les deux sous-réseaux sont utiles puisqu'on cherche la translation d'une petite sous-région dans une grande sous-région. Étant donné qu'on travaille avec deux spectres distincts, échanger les domaines des petites et grandes sous-régions fait du sens pour permettre une validation des prédictions des deux sous-réseaux. Or, dans la deuxième architecture, les deux sous-régions sont de la même taille, donc avoir cette même sorte de validation n'était pas possible.

Une autre observation est que notre deuxième méthode est beaucoup plus performante que notre première. En effet, pour exactement les mêmes données de test, on note une différence de 12.7% entre les résultats du premier article (77.9%) et du deuxième (90.6%). Nous croyons qu'il existe plusieurs facteurs pouvant expliquer cette différence.

La première se trouve dans les architectures des RNCs, plus spécifiquement dans la partie qui extrait les caractéristiques. Dans le premier article, le réseau développé pour extraire les caractéristiques utilisait des filtres de convolutions 7×7 alors que le deuxième utilisait des filtres 5×5. En plus de nécessiter un plus grand nombre de paramètres, les filtres de convolution de plus grande taille peuvent négliger des petits détails de la scène. Dans le cas d'estimation de disparités pour des silhouettes humaines, ces petits détails peuvent s'avérer très importants, ce qui peut expliquer en partie la différence de performances. De plus, le plus grand nombre de paramètres peut aussi causer un peu de sur-apprentissage, ce qui fait que les performances sur l'ensemble de test peuvent diminuer par rapport à l'ensemble de validation. Nous avons remarqué que pour le premier article, il y avait une différence de $5-6\%$ entre les résultats de validation et les résultats de test. Cette différence était beaucoup moins importante pour le deuxième article $(1-2\%)$. Ceci nous laisse croire que la deuxième architecture développée possède une meilleure capacité à généraliser que la première.

Un autre facteur qui pourrait expliquer la différence entre les résultats se trouve également dans l'extraction des caractéristiques. Dans le premier article, la dimension finale des canaux est de 64 alors qu'elle est de 256 pour le deuxième. Dans le premier cas, on diminue la résolution spatiale grâce aux convolutions, mais on atteint le maximum en profondeur des

canaux assez tôt, ce qui fait que certaines caractéristiques importantes ont peut-être été ignorées dû au manque d'espace pour les représenter. Ce problème n'apparaît pas dans le deuxième article, puisqu'au fur et à mesure qu'on réduit la taille des cartes de caractéristiques, on augmente progressivement la profondeur des canaux jusqu'à obtenir une profondeur de 256. Ceci nous permet donc, pour un même vecteur caractéristique d'une sous-région de même taille, d'avoir quatre fois plus d'information pour permettre au réseau d'apprendre à établir les correspondances correctement.

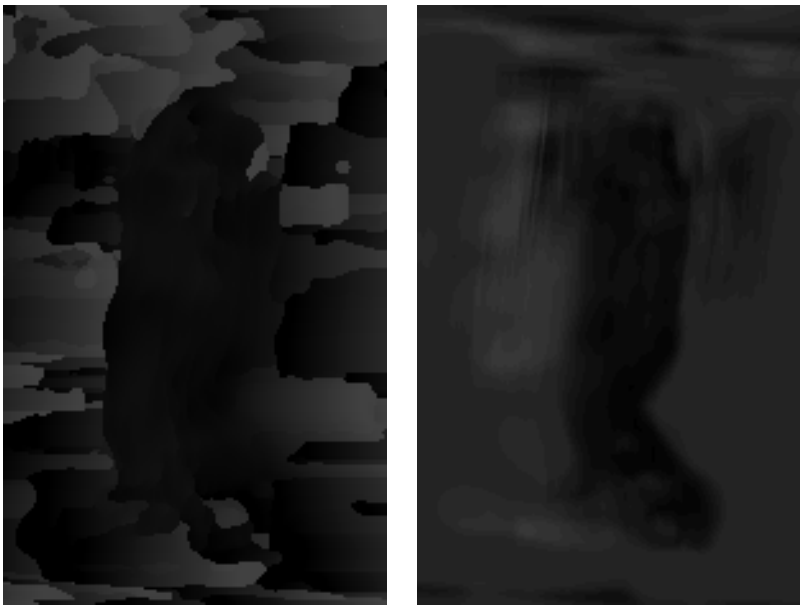
Le troisième facteur est le nombre de données disponibles lors de l'entraînement des réseaux. Dans les deux articles, les mêmes données brutes ont été utilisées (mêmes images de base). Or, l'augmentation de données a été modifiée pour le deuxième article, ce qui fait qu'il y avait un beaucoup plus grand nombre de points d'entraînements de disponible. Avec la séparation en plusieurs blocs (*folds*), l'article 1 avait en moyenne 145 000 points dans l'ensemble d'entraînement, alors que l'article numéro deux en avait en moyenne 510 000 points d'entraînement, soit plus que trois fois plus. Selon nous, ceci est un autre facteur pouvant expliquer la différence des résultats. En effet, si on regarde les résultats du premier article par bloc, on observe une bonne différence entre ceux-ci. Le meilleur résultat (bloc 1, 83.9%) est celui qui a le plus de données d'entraînement, alors que les deux autres sont 5% et 11% inférieurs, et avec moins de points d'entraînement. Dans le cas du deuxième article, il y a une différence de 1.5% entre le meilleur bloc et le pire, ce qui laisse présager qu'avoir plus de données pour tous les blocs permet d'atténuer les différences de performances entre ceux-ci, et qui permet d'avoir de meilleurs résultats globalement. Effectivement, des tests montrent que l'entraînement de l'article 1 avec les données de l'article 2 réduit la différence de performance entre nos deux méthodes d'approximativement 12.7% à environ 9%.

Les figures 6.1, 6.3, 6.4 montrent des résultats qualitatifs des cartes de disparités obtenues par nos articles pour des scènes de une, trois et cinq personnes. Ceci nous permet de continuer la comparaison des deux méthodes, mais aussi de voir quelques faiblesses qui sont présentes. Pour les deux méthodes, nous avons utilisé les masques de segmentation fournis par l'ensemble de données pour isoler les silhouettes humaines de la scène. Puisque les points d'entraînements proviennent des pixels sur les silhouettes, nos méthodes ont plus de difficultés à estimer des disparités dans le décor de la scène. De plus, étant donné qu'il est impossible de voir des détails à l'exception des humains dans l'image LWIR, les disparités prédites varient beaucoup d'un pixel à l'autre lorsqu'ils ne sont pas sur les silhouettes. La figure 6.2 montre des exemples du sujet de la première paire d'images (figure 6.1) avec une partie du décor. La différence entre les silhouettes segmentées (figure 6.1 et les silhouettes non segmentées (figure 6.2) montre qu'à l'extérieur de la silhouette de la personne, les disparités ne sont pas aussi précises, et que sans le masque de segmentation, les formes des personnes apparaissent beaucoup plus

Figure 6.1 Images RGB et LWIR (première rangée) avec les cartes de disparités segmentées de l'article 1 et 2 (deuxième rangée). Les pixels foncés correspondent à des plus petites disparités. Il est conseillé de zoomer sur les résultats pour bien voir les valeurs de disparités.

grosses qu'elles le sont réellement. Nous croyons que le manque de précision dans les formes des sujets est causé par un manque de pixels avec annotations aux frontières. En effet, la plupart des données d'entraînement sont des points à l'intérieur des frontières de la personne, ce qui fait que les prédictions des pixels aux frontières, déjà une difficulté due à la différence qu'il y a entre RGB et LWIR à ce niveau, sont moins précises. Par contre, à l'aide de la segmentation, ce problème est moins présent étant donné que les frontières sont alors définies par la qualité de la segmentation.

En observant les paires d'images, on remarque que plus il y a de sujets dans la scène, moins les cartes de disparités semblent être de bonne qualité. En effet, à la figure 6.1, on voit que les cartes de disparités sont plutôt uniformes, à l'exception du pied du sujet dans la carte produite par l'article 1. Or, dans un cas de trois personnes, tel que montré à la figure 6.3, on voit déjà qu'il y a un peu plus de bruit dans les disparités, spécifiquement dans le sujet le plus à gauche. Dans une image avec cinq personnes, on voit encore ce problème comme on l'observe à la figure 6.4. Dans tous les cas, on remarque que les disparités sur les personnes sont beaucoup plus uniformes dans l'article 2. Ceci est conséquent avec les



(a) (b)

Figure 6.2 Cartes de disparités sans les masques de segmentations pour la région autour du sujet présent dans la figure 6.1. (a) première méthode, (b) deuxième méthode. Les pixels foncés correspondent à des plus petites disparités. On observe beaucoup plus de bruit dans la première méthode.

meilleures performances obtenues sur les ensembles de données par la deuxième méthode. On remarque toutefois que les deux méthodes ont des difficultés avec les occlusions. Le sujet caché derrière la personne en rouge à la figure 6.4 n'apparaît pas à une profondeur différente alors qu'il le devrait. Dans les deux cas, nos méthodes échouent, mais ceci est probablement dû au fait qu'il s'agit d'une occlusion presque complète.

Avant l'élaboration du deuxième article, nous avons essayé différents changements à l'architecture du premier article. Le plus intéressant de ceux-ci est le changement du produit de corrélation par une couche convolutive qui pouvait apprendre à corréler les éléments les plus importants. Le principe était de prendre le vecteur provenant de la petite sous-région, et de le répéter plusieurs fois jusqu'à l'obtention d'un volume de caractéristiques de même taille que celui provenant de la grande sous-région. Par la suite, on concaténait les deux volumes de sorte que le vecteur (répété plusieurs fois) soit aligné avec les différents vecteurs correspondant à des disparités possibles du volume. Ensuite, ce bloc de caractéristique était propagé dans une couche de convolution qui pouvait apprendre les relations entre le vecteur de la petite sous-région et les vecteurs du volume de la grande à chaque disparité possible. Or, les résultats obtenus n'étaient pas supérieurs à ceux de l'article (légèrement inférieurs), donc

Figure 6.3 Images RGB et LWIR (première rangée) avec les cartes de disparités segmentées de l'article un et deux (deuxième rangée). Les pixels foncés correspondent à des plus petites disparités. Avec trois personnes dans la scène, on remarque que la méthode 1 a un peu plus de difficultés. Il est conseillé de zoomer sur les résultats pour bien voir les valeurs de disparités.

l'idée à été abandonnée. De plus, ceci ajoutait du temps d'exécution par rapport au produit de corrélation qui est très rapide (reviens à un produit vecteur-matrice).

Nous avons aussi essayé d'adapter un réseau quadruplet similaire au réseau de Aguilera *et al.* [11]. On utilisait deux paires de sous-régions correspondantes (A et B) à partir desquelles on obtenait des vecteurs de caractéristiques pour chaque image. On avait donc un vecteur de caractéristiques de la sous-région RGB de la paire A, un autre de la sous-région LWIR également de la paire A, un troisième vecteur provenant d'une autre sous-région RGB (paire B), et le dernier vecteur pour la sous-région LWIR aussi de la paire B. Ceci nous permettait de construire une fonction de coût avec comme objectif de maximiser le coût des deux paires correspondantes et de minimiser les quatre autres paires possibles qui ne correspondent pas. Le problème de cette approche est qu'elle était mal adaptée aux données avec lesquelles on travaillait. Dans l'article de Aguilera *et al.* [11], les auteurs utilisent des images RGB et NIR qui ont beaucoup d'information en commun. Or, dans notre cas, avec des images RGB et LWIR, seulement certaines régions ont des points communs évidents, ce qui faisait que le

Figure 6.4 Images RGB et LWIR (première rangée) avec les cartes de disparités segmentées de l'article un et deux (deuxième rangée). Les pixels foncés correspondent à des plus petites disparités. Remarquons qu'un sujet est presque complètement caché par un autre, ce qui n'est pas réflété dans aucune des cartes de disparités. Il est conseillé de zoomer sur les résultats pour bien voir les valeurs de disparités.

réseau avait de la difficulté à apprendre, et que les résultats obtenus tournaient près des 55%. Une autre explication pourquoi cette méthode ne donnait pas les résultats escomptés provient du fait qu'il ne s'agit pas de la même tâche. Aguilera *et al.* [11] avait pour but de développer un descripteur pour des images RGB et NIR, ce qui fait que l'ensemble de données qu'ils ont utilisé avait des annotations à des points détectés par le descripteur SIFT [48]. Ceci fait que les pixels à partir desquelles ils s'entraînaient étaient déjà discriminants (choisis par SIFT), contrairement à nous qui utilisions un ensemble de données pour la stéréoscopie. Pour ces raisons et les mauvais résultats, cette idée a également été abandonnée.

CHAPITRE 7 CONCLUSION

Dans ce mémoire, nous avons présenté deux nouvelles méthodes permettant d'estimer les disparités entre des images multispectrales RGB et LWIR. Plusieurs facteurs rendent ce problème difficile (occlusions, pixels aux frontières, objets froids), mais la principale difficulté provient du fait que les spectres utilisés (RGB et LWIR) ont très peu d'information en commun. Classiquement résolues avec des descripteurs développés par des humains, nos méthodes se basent sur l'apprentissage profond, plus précisément les RNCs, afin d'apprendre à établir des correspondances entre les paires d'images multispectrales.

7.1 Synthèse des travaux

Voici la liste des contributions de nos deux méthodes :

— Nous avons tout d'abord proposé une nouvelle architecture basée sur deux réseaux siamois prenant en entrées deux paires de sous-régions provenant d'images multispectrales. Utilisant un RNC pour extraire des caractéristiques, nous effectuons un produit de corrélation dans l'espace des caractéristiques pour obtenir une distribution de probabilités des disparités possibles pour chaque réseau siamois de l'architecture. On combine leurs résultats pour être plus confiant en la prédiction du réseau et avoir une méthode plus robuste.

— Nous avons ensuite proposé une autre architecture profonde pour estimer les disparités entre des images multispectrales. Celle-ci utilise un RNC plus profond pour extraire de meilleures caractéristiques pour les images RGB et LWIR. Les vecteurs résultants sont alors joints de deux manières différentes : par corrélation et par concaténation. Ceci ajoute de la robustesse à notre réseau, qui peut apprendre à effectuer les mêmes correspondances de deux manières différentes. Le réseau donne une probabilité que les sous-régions d'entrées soient les mêmes ou non.

— Nous avons effectué plusieurs expériences sur les bases de données LITIV 2014 [1] et LITIV 2018 [4] montrant, pour la première méthode, que les RNCs pouvaient obtenir des résultats compétitifs par rapport aux descripteurs classiques. Dans la deuxième méthode, nous avons démontré qu'ils pouvaient obtenir des résultats supérieurs pour l'estimation des disparités.

7.2 Limitations de la solution proposée

Nos deux méthodes ont été entraînées sur des données dites éparses. Ceci signifie que le nombre d'annotations dans les paires d'images est très bas. En observant les cartes de disparités présentées au chapitre 6, on voit que nos méthodes ont des difficultés à reproduire les formes exactes des silhouettes des sujets de la scène. De plus, surtout dans le cas de la première méthode, les disparités aux pixels à l'extérieur des personnes sont souvent erronées.

7.3 Améliorations futures

Il y a deux améliorations futures que nous jugeons pertinentes pour améliorer les résultats présentés dans ce mémoire. Tout d'abord, utiliser une autre tâche connexe comme la segmentation pourrait aider le problème que nos méthodes ont avec la forme des silhouettes. En effet, avec une segmentation des personnes de la scène, nous pourrions incorporer ces informations dans le RNC afin d'améliorer ses performances pour les pixels aux frontières. Ce genre d'approche est utilisé dans la vision stéréoscopique visible [10, 32, 47], donc le défi serait de l'adapter au domaine infrarouge.

La deuxième amélioration future serait par rapport aux données. Tel que mentionné dans le mémoire, il y a très peu de bases de données avec des images RGB et LWIR, ce qui fait qu'entraîner des RNC est plus compliqué. De plus, ces bases de données ne sont pas annotées de façon dense, et le nombre de points contenus dans celles-ci est plutôt bas. Créer une nouvelle base de données avec des annotations denses pourrait grandement aider les méthodes stéréoscopiques travaillant dans des domaines de spectres différents. Ceci permettrait le développement d'architectures bout-à-bout, qui sont supérieures aux méthodes par sous-régions, et qui obtiennent des cartes de disparités de plus grande qualité. La vision stéréoscopique profiterait grandement d'une base de données comparable à KITTI [59] ou à Sceneflow [38]. À défaut de créer de nouvelles bases de données, il serait intéressant d'explorer des méthodes non-supervisées qui sont capables d'obtenir des résultats intéressants malgré l'absence d'annotations, telles que les méthodes de Liang *et al.* [57] et Zhi *et al.* [12].

RÉFÉRENCES

[1] G.-A. Bilodeau *et al.*, "Thermal-visible registration of human silhouettes : A similarity measure performance evaluation," *Infrared Physics & Technology*, vol. 64, nº. C, p. 79–86, 2014.

[2] S. Ioffe et C. Szegedy, "Batch normalization : Accelerating deep network training by reducing internal covariate shift," dans *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 448–456. [En ligne]. Disponible : http://dl.acm.org/citation.cfm?id=3045118.3045167

[3] V. Nair et G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," dans *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. USA : Omnipress, 2010, p. 807–814. [En ligne]. Disponible : http://dl.acm.org/citation.cfm?id=3104322.3104425

[4] P.-L. St-Charles, G.-A. Bilodeau et R. Bergevin, "Online mutual foreground segmentation for multispectral stereo videos," *International Journal of Computer Vision*, Jan 2019. [En ligne]. Disponible : https://doi.org/10.1007/s11263-018-01141-5

[5] M. Brown et S. Süsstrunk, "Multispectral SIFT for scene category recognition," dans *Computer Vision and Pattern Recognition (CVPR11)*, Colorado Springs, June 2011, p. 177–184.

[6] J. Zbontar et Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[7] F. Guney et A. Geiger, "Displets : Resolving stereo ambiguities using object knowledge," dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] W. Luo, A. Schwing et R. Urtasun, "Efficient deep learning for stereo matching," dans *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] A. Kendall *et al.*, "End-to-end learning of geometry and context for deep stereo regression," dans *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[10] Z. Wu *et al.*, "Semantic stereo matching with pyramid cost volumes," dans *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[11] C. A. Cristhian A. Aguilera, Angel D. Sappa et R. Toledo, “Cross-spectral local descriptors via quadruplet network,” dans *Sensors*, 2017, p. 17(4) :873.

[12] T. Zhi *et al.*, “Deep material-aware cross-spectral stereo matching,” dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] D. Scharstein *et al.*, “High-resolution stereo datasets with subpixel-accurate ground truth,” dans *Pattern Recognition*, X. Jiang, J. Hornegger et R. Koch, édit. Cham : Springer International Publishing, 2014, p. 31–42.

[14] D. Scharstein et R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *International Journal of Computer Vision*, vol. 47, n°. 1, p. 7–42, Apr 2002. [En ligne]. Disponible : https://doi.org/10.1023/A:1014573219977

[15] L. H. Matthies, T. Kanade et R. Szeliski, “Kalman filter-based algorithms for estimating depth from image sequences.” *International Journal of Computer Vision*, vol. 3, n°. 3, p. 209–238, 1989. [En ligne]. Disponible : https://doi.org/10.1007/BF00133032

[16] T. Kanade *et al.*, “Development of a video-rate stereo machine,” dans *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 3, 1995, p. 95–100 vol.3.

[17] D. Scharstein et R. Szeliski, “Stereo matching with nonlinear diffusion,” *International Journal of Computer Vision*, vol. 28, n°. 2, p. 155–174, June 1998. [En ligne]. Disponible : https://www.microsoft.com/en-us/research/publication/stereo-matching-with-nonlinear-diffusion/

[18] T. W. Ryan, R. T. Gray et B. R. Hunt, “Prediction Of Correlation Errors In Stereo-Pair Images,” *Optical Engineering*, vol. 19, n°. 3, p. 312 – 322, 1980. [En ligne]. Disponible : https://doi.org/10.1117/12.7972515

[19] H. H. Baker et T. O. Binford, “Depth from edge and intensity based stereo,” dans *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’81. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1981, p. 631–636.

[20] A. Bobick et S. Intille, “Large occlusion stereo,” *International Journal of Computer Vision*, vol. 33, p. 181–200, 09 1999.

[21] T. Kanade et M. Okutomi, “A stereo matching algorithm with an adaptive window : theory and experiment,” dans *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, p. 1088–1095 vol.2.

[22] D. Terzopoulos, “Regularization of inverse visual problems involving discontinuities,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, n°. 4, p. 413–424, 1986.

[23] Y. Ohta et T. Kanade, “Stereo by intra- and inter-scanline search using dynamic programming,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, nº. 2, p. 139–154, 1985.

[24] D. Marr et T. Poggio, “Cooperative computation of stereo disparity,” *Science (New York, N.Y.)*, vol. 194, p. 283–7, 11 1976.

[25] M. Shimizu et M. Okutomi, “Precise sub-pixel estimation on area-based matching,” dans *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, 2001, p. 90–97 vol.1.

[26] S. Roy et I. J. Cox, “A maximum-flow formulation of the n-camera stereo correspondence problem,” dans *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, p. 492–499.

[27] S. Birchfield et C. Tomasi, “A pixel dissimilarity measure that is insensitive to image sampling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, nº. 4, p. 401–406, 1998.

[28] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, nº. 2, p. 328–341, Feb 2008.

[29] P. Viola et W. M. Wells, “Alignment by maximization of mutual information,” dans *Proceedings of IEEE International Conference on Computer Vision*, June 1995, p. 16–23.

[30] R. Zabih et J. Woodfill, *Non-parametric local transforms for computing visual correspondence*, 04 2006, vol. 801, p. 151–158.

[31] N. Dalal et B. Triggs, “Histograms of oriented gradients for human detection,” dans *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, June 2005, p. 886–893 vol. 1.

[32] M. Bleyer *et al.*, “Object stereo — joint stereo matching and object segmentation,” dans *CVPR 2011*, June 2011, p. 3081–3088.

[33] D. Scharstein et C. Pal, “Learning conditional random fields for stereo,” dans *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, p. 1–8.

[34] Z. Chen *et al.*, “A deep visual correspondence embedding model for stereo matching costs,” dans *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, p. 972–980.

[35] H. Park et K. M. Lee, “Look wider to match image patches with convolutional neural networks,” *CoRR*, vol. abs/1709.06248, 2017. [En ligne]. Disponible : http://arxiv.org/abs/1709.06248

[36] A. Shaked et L. Wolf, “Improved stereo matching with constant highway networks and reflective loss,” *arXiv preprint arxiv :1701.00165*, 2016.

[37] K. He *et al.*, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [En ligne]. Disponible : http://arxiv.org/abs/1512.03385

[38] N. Mayer *et al.*, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” dans *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, arXiv :1512.02134. [En ligne]. Disponible : http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16

[39] A. Dosovitskiy *et al.*, “Flownet : Learning optical flow with convolutional networks,” dans *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[40] J. Pang *et al.*, “Cascade residual learning : A two-stage convolutional neural network for stereo matching,” *CoRR*, vol. abs/1708.09204, 2017. [En ligne]. Disponible : http://arxiv.org/abs/1708.09204

[41] J.-R. Chang et Y.-S. Chen, “Pyramid stereo matching network,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, p. 5410–5418.

[42] K. He *et al.*, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” dans *Computer Vision – ECCV 2014*, D. Fleet *et al.*, édit. Cham : Springer International Publishing, 2014, p. 346–361.

[43] A. Seki et M. Pollefeys, “Sgm-nets : Semi-global matching with neural networks,” dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] F. Zhang *et al.*, “Ga-net : Guided aggregation net for end-to-end stereo matching,” dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[45] X. Guo *et al.*, “Group-wise correlation stereo network,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, p. 3273–3282.

[46] X. Song *et al.*, “Edgestereo : A context integrated residual pyramid network for stereo matching,” dans *Computer Vision – ACCV 2018*, C. Jawahar *et al.*, édit. Cham : Springer International Publishing, 2019, p. 20–35.

[47] G. Yang *et al.*, “Segstereo : Exploiting semantic information for disparity estimation,” dans *The European Conference on Computer Vision (ECCV)*, September 2018.

[48] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, n°. 2, p. 91–110, nov. 2004. [En ligne]. Disponible : https://doi.org/10.1023/B:VISI.0000029664.99615.94

[49] A. Torabi et G.-A. Bilodeau, “Local self-similarity-based registration of human rois in pairs of stereo thermal-visible videos,” *Pattern Recognition*, vol. 46, n°. 2, p. 578 –

589, 2013. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0031320312003445

[50] A. Alahi, R. Ortiz et P. Vandergheynst, "Freak : Fast retina keypoint," dans *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, p. 510–517.

[51] M. Calonder *et al.*, "Brief : Binary robust independent elementary features," dans *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos et N. Paragios, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, p. 778–792.

[52] E. B. Baruch et Y. Keller, "Multimodal matching using a hybrid convolutional neural network," *CoRR*, vol. abs/1810.12941, 2018. [En ligne]. Disponible : http://arxiv.org/abs/1810.12941

[53] M. Pistarelli, A. Sappa et R. Toledo, "Multispectral stereo image correspondence," 08 2013, p. 217–224.

[54] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[55] C. Aguilera *et al.*, "Multispectral image feature points," *Sensors (Basel, Switzerland)*, vol. 12, p. 12 661–72, 12 2012.

[56] C. A. Aguilera *et al.*, "Learning cross-spectral similarity measures with deep convolutional neural networks," dans *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016, p. 267–275.

[57] M. Liang *et al.*, "Unsupervised cross-spectral stereo matching by learning to synthesize," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, p. 8706–8713, 07 2019.

[58] J.-Y. Zhu *et al.*, "Unpaired image-to-image translation using cycle-consistent adversarial networks," dans *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[59] M. Menze et A. Geiger, "Object scene flow for autonomous vehicles," dans *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] J. Redmon et A. Farhadi, "YOLO9000 : better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [En ligne]. Disponible : http://arxiv.org/abs/1612.08242

[61] S. Ren *et al.*, "Faster R-CNN : towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [En ligne]. Disponible : http://arxiv.org/abs/1506.01497

[62] A. Krizhevsky, I. Sutskever et G. E. Hinton, "Imagenet classification with deep convolutional neural networks," dans *Advances in Neural Information Processing Systems 25*, F. Pereira *et al.*, édit. Curran Associates,

Inc., 2012, p. 1097–1105. [En ligne]. Disponible : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[63] C. Szegedy *et al.*, "Going deeper with convolutions," dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, p. 1–9.

[64] Y. Xiang, A. Alahi et S. Savarese, "Learning to track : Online multi-object tracking by decision making," dans *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, p. 4705–4713.

[65] S. Yun *et al.*, "Action-decision networks for visual tracking with deep reinforcement learning," dans *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, p. 1349–1358.

[66] Z. Jie *et al.*, "Left-right comparative recurrent model for stereo matching," dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[67] J. Zbontar et Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, p. 1–32, 2016.

[68] G. Bilodeau, A. Torabi et F. Morin, "Visible and infrared image registration using trajectories and composite foreground images," *Image and Vision Computing*, vol. 29, n°. 1, p. 41 – 50, 2011. [En ligne]. Disponible : http://www.sciencedirect.com/science/article/pii/S0262885610001101

[69] A. Torabi, M. Najafianrazavi et G. Bilodeau, "A comparative evaluation of multimodal dense stereo correspondence measures," dans *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Sept 2011, p. 143–148.

[70] E. Shechtman et M. Irani, "Matching local self-similarities across images and videos," dans *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, p. 1–8.

[71] N. Srivastava *et al.*, "Dropout : A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, p. 1929–1958, 2014. [En ligne]. Disponible : http://jmlr.org/papers/v15/srivastava14a.html

[72] J. Duchi, E. Hazan et Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, p. 2121–2159, juill. 2011. [En ligne]. Disponible : http://dl.acm.org/citation.cfm?id=1953048.2021068

[73] C. Zhang *et al.*, "Meshstereo : A global stereo model with mesh alignment regularization for view interpolation," dans *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 2057–2065.

[74] X. Chen *et al.*, "3d object proposals for accurate object class detection," dans *Advances in Neural Information Processing Systems 28*, 2015, p. 424–432.

[75] D. Quan *et al.*, “Afd-net : Aggregated feature difference learning for cross-spectral image patch matching,” dans *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[76] A. Geiger, P. Lenz et R. Urtasun, “Are we ready for autonomous driving ? the kitti vision benchmark suite,” dans *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[77] D.-A. Beaupre et G.-A. Bilodeau, “Siamese cnns for rgb-lwir disparity estimation,” dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[78] E. Hoffer et N. Ailon, “Deep metric learning using triplet network,” dans *Similarity-Based Pattern Recognition*, A. Feragen, M. Pelillo et M. Loog, édit. Cham : Springer International Publishing, 2015, p. 84–92.

[79] H. Hirschmuller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, n°. 2, p. 328–341, Feb 2008.

[80] X. Song *et al.*, “Edgestereo : A context integrated residual pyramid network for stereo matching,” dans *Computer Vision – ACCV 2018*, C. Jawahar *et al.*, édit. Cham : Springer International Publishing, 2019, p. 20–35.

[81] A. Paszke *et al.*, “Pytorch : An imperative style, high-performance deep learning library,” dans *Advances in Neural Information Processing Systems 32*, H. Wallach *et al.*, édit. Curran Associates, Inc., 2019, p. 8024–8035. [En ligne]. Disponible : http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[82] D. Kingma et J. Ba, “Adam : A method for stochastic optimization,” 2014, cite arxiv :1412.6980Comment : Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [En ligne]. Disponible : http://arxiv.org/abs/1412.6980