Deep learning for light field view synthesis from monocular and a very sparse set of input views

Simon Evain

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Simon EVAIN

Deep learning for light field view synthesis from monocular and a very sparse set of input views

Thèse présentée et soutenue à Rennes, le 12 juillet 2021
INRIA Rennes - Bretagne Atlantique

Rapporteurs avant soutenance :

Gabriele FACCIOLO	Professor, ENS
Frédéric CHAMPAGNAT	Researcher, ONERA

Composition du Jury :

Examinateurs :	Eric MARCHAND	Professor, Université Rennes 1
	Frédéric CHAMPAGNAT	Researcher, ONERA
	Gabriele FACCIOLO	Professor, ENS
	Sebastian KNORR	Professor, Ernst-Abbe University
Dir. de thèse :	Christine GUILLEMOT	Research Director, INRIA Rennes

Table of Contents

Résumé en français

Contexte

Les techniques de réalité virtuelle et de réalité augmentée se développent aujourd'hui massivement dans nos sociétés. La raison de cet engouement est aisément compréhensible : elles proposent en effet aux utilisateurs des expériences plus immersives, et spectaculaires qui trouvent un grand intérêt, notamment auprès des férus de produits culturels tels que le jeu vidéo ou le cinéma. Ainsi, en France, la vente annuelle de casques de réalité virtuelle a par exemple été multipliée par 5.5 entre 2015 et 2019, passant de 60 000 à près de 330 000 unités vendues (*Source : IDATE DigiWorld*). On peut dire la même chose des marchés de la réalité augmentée et de la réalité mixte, porteurs et en forte croissance notamment grâce aux investissements colossaux réalisés par les GAFAM.

La pandémie de Covid-19, qui sévit partout dans le monde entier à l'heure de la réalisation de cette thèse, est également annonciatrice probable de mutations profondes à venir. Si le télétravail et la distanciation sociale se sont imposés comme de nouvelles normes sociales lors de ces derniers mois, de nombreux instituts et centres ont également dû repenser leur organisation et leur offre à un public contraint de rester à son domicile. En premier lieu les acteurs culturels, privés de recettes, qui ont dû et su s'adapter à la nouvelle donne en proposant des services de visites de musées à distance, ou de spectacles vivants en diffusion sur Internet. De même, de nombreuses entreprises, telles que les agences immobilières, ont su faire face aux restrictions en s'engageant dans le développement de technologies innovantes, telles que la visite virtuelle d'appartements. Si ces investissements ont bien entendu été accélérés par un contexte dramatique, il ne fait guère de doutes qu'ils survivront, au moins en partie, à la pandémie, et constituent une première pierre du monde de l'après Covid. Il y a fort à parier que cette crise aura des conséquences profondes relatives au développement de ces technologies sur le long terme.

Tous ces développements reposent, d'un point de vue technique, sur 2 piliers essentiels. D'abord, ces technologies sont aujourd'hui construites sur des méthodes d'*intelligence arti-*

ficielle (ou *apprentissage profond*), qui trouvent leur efficacité de l'analyse d'une multitude de données. Ces données sont omniprésentes aujourd'hui, notamment grâce à Internet et aux réseaux sociaux. Ce n'est ainsi pas une coïncidence si on retrouve les GAFAM en meneurs actuels du mouvement : les autres services qu'ils dirigent et détiennent par ailleurs sont utiles pour la collecte de données et d'images permettant la mise au point de ces technologies innovantes. Ainsi en est-il de Facebook à la pointe de la réalité virtuelle grâce à ses casques Oculus, de Google développant aujourd'hui des techniques de capture en réalité virtuelle basée sur les champs de lumière au niveau de l'état de l'art, ou de Microsoft et son Hololens, meneur du domaine de la réalité mixte.

Un autre élément sur lequel s'appuient ces technologies est la capacité à capturer différents points de vue d'une même scène, pour la reconstruire ensuite efficacement dans son intégralité. Cette contrainte est significative, dans la mesure où elle limite les possibilités de capture à des équipements plutôt sophistiqués et difficiles à acquérir en 2021 pour un particulier. Si ce constat est indéniable, il convient tout de même de le nuancer, en évoquant l'omniprésence des smartphones aujourd'hui. De plus en plus vendus avec plusieurs capteurs photo présents simultanément, ils peuvent constituer dans les années à venir une réponse intéressante aux problèmes de complexité d'équipement. Ils restent toutefois toujours limités par la taille désirée du smartphone, ne permettant pas de traiter des scènes prises de points de vue radicalement distants.

Nos travaux décrits ci-après s'inscrivent pleinement dans ce contexte et dans cette dynamique.

Motivations et objectifs

Dans cette thèse, nous nous sommes intéressés en particulier à un certain type de données multivues : les champs de lumière. Nous y trouvons un intérêt particulier car ces données sont utilisées d'une part dans le champ industriel (par des entreprises telles que Raytrix), d'autre part pour l'amélioration des technologies en réalité virtuelle et augmentée (voir la démonstration de Google *Welcome to Light Fields*).

Sur le plan formel, on peut présenter ces champs de lumière comme la mesure de l'intensité de l'ensemble des rayons de lumière circulant par l'ensemble des points d'une même

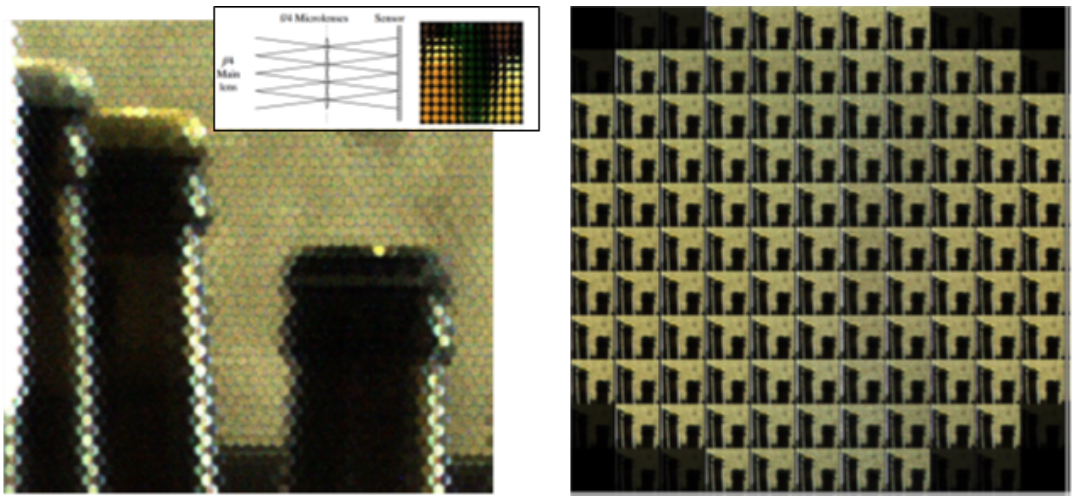

FIGURE 1 – Double représentation des champs de lumière : sur la gauche, la représentation par micro-lentilles ; sur le droite, la représentation par sous-images.

scène. Ils sont décrits mathématiquement en particulier par la *fonction plénoptique*, qui associe à chaque paramétrisation de rayon lumineux au sein d'une scène une intensité (la *radiance*). La connaissance fine de la trajectoire de ces différents rayons lumineux ouvre la voie à de nombreuses applications utiles, spécifiques aux champs de lumière. D'une part, la technologie plénoptique nous permet de générer une collection de points de vues, dans laquelle il est possible de simuler un déplacement (voir figure 1). D'autre part, elle nous donne davantage d'éléments sur la géométrie de la scène, et est un outil utile pour estimer la profondeur des différents objets à partir d'une seule capture. Enfin, d'autres applications, telles que la *refocalisation numérique* (refocaliser après capture) sont également accessibles. Du fait de sa meilleure capacité à capturer les différentes trajectoires de rayons, elle peut également constituer un atout significatif dans la capture des contenus de réalité augmentée et virtuelle.

Les champs de lumière ne peuvent être capturés en utilisant un appareil photo conventionnel : en effet, pour de tels appareils, l'information obtenue lors de la capture photo représente la somme des rayons lumineux ayant transité dans l'appareil et ayant convergé sur un pixel donné du photocapteur. Il est donc impossible, pendant ou après la capture, de faire la distinction entre les différents rayons lumineux qui contribuent à l'intensité de ce pixel. Il est alors impératif d'utiliser un équipement spécifique pour la capture de tels dispositifs.

FIGURE 2 – A gauche, la caméra plénoptique Lytro Illum (source : *Wikipedia*), à droite la matrice de caméras Stanford.

Pour cela, deux types de dispositifs existent, comme illustré en figure 2 : d'abord, les *caméras plénoptiques*, similaires aux appareils photo traditionnels, à la différence qu'une grille de microlentilles est disposée devant le photocapteur. Ainsi, les rayons lumineux se verront réfractés par les microlentilles avant d'atteindre le capteur, et seront donc discriminés en fonction de leur direction d'incidence. De tels appareils sont confrontés à ce que l'on appelle le *compromis spatio-angulaire* : puisque le nombre de pixels capturable par notre capteur est par essence limité, un compromis doit être réalisé entre la résolution des images que l'on souhaite synthétiser, et le nombre de rayons lumineux que l'on souhaite être en mesure de traiter. Ceci conduit habituellement les caméras plénoptiques à soit nécessiter des photocapteurs à très haute résolution ; amenant des prix de vente prohibitifs pour le grand public. Ou à réduire la résolution des images produites, créant de ce fait des différences de qualité d'image significatives et problématiques pour les amateurs de photographie.

Un autre type de dispositif existe : les *matrices de caméras*. Il s'agit de dispositifs regroupant un grand nombre de caméras au sein d'une même structure. Si elles évitent l'essentiel des écueils des caméras plénoptiques, elles ont un défaut évident pour un usage grand public : elles sont extrêmement encombrantes, et de plus difficiles à régler, notamment car la synchronicité des différentes caméras est essentielle.

Un des objectifs de cette thèse est de s'intéresser à cette problématique en cherchant à répondre à la question suivante :

Dans quelle mesure et sous quelles conditions est-il possible de reconstruire un champ

de lumières à partir de captures avec une unique caméra 2D ?

De cette problématique émanent plusieurs contraintes, puisque l'on souhaite en effet restreindre au maximum les exigences du point de vue utilisateur. Ainsi, on souhaite que les méthodes développées aient une *barrière d'entrée* aussi basse que possible pour les utilisateurs. On souhaite également que les algorithmes et méthodes proposés puissent être utilisables pour des smartphones, ce qui implique donc une complexité à réduire au maximum.

Lors de cette thèse, nous envisageons le problème par le biais de la synthèse de vues : nous cherchons à mener une reconstruction efficace d'un champ de lumière, en utilisant un nombre extrêmement restreint de vues d'entrée et cherchons à générer le reste du champ de lumière ainsi. Là où en théorie, 4 vues bien choisies sont habituellement nécessaires pour reconstruire fidèlement un champ de lumière (puisque par ce choix, l'ensemble des points tridimensionnels constituant la scène peuvent être retrouvés), nous nous plaçons volontairement dans un cadre plus accessible pour un utilisateur lambda, mais beaucoup plus complexe sur le plan technique, en prenant pour référence encore moins de vues d'entrée (une seule dans les chapitres 3 et 4, de 1 à 4 dans le chapitre 5). A l'instant du test, nous sommes donc confrontés à un manque d'informations pour pouvoir regénérer fidèlement le champ de lumière.

Comment parvenir alors à travailler sur ces informations sans y avoir accès au moment du test ? Pour faire cela, nous utilisons des techniques d'*apprentissage profond*. L'idée essentielle est qu'au moment de l'apprentissage, nos réseaux de neurone devront apprendre les informations qu'il leur manquera nécessairement au moment du test, pour être en mesure de traiter le cas test efficacement. Il est également bon de noter que pour le cas où une seule image est utilisée en entrée, cette problématique est d'autant plus notable.

Dans notre processus d'apprentissage profond, nous souhaitons donc adopter une complexité aussi faible que possible, et nous cherchons à réduire au maximum tant le temps d'apprentissage que la quantité de données d'entraînement nécessaire.

Nos contributions

Nous présentons dans les chapitres qui suivent nos réponses à la problématique fixée. Dans un premier temps (**chapitre 1**), nous nous penchons sur la question des champs de lumière, et nous en décrivons leurs caractéristiques, leur importance et leurs limitations. Ceci permet de comprendre la pertinence des méthodes déployées lors de cette thèse, et donne une idée des travaux déjà réalisés dans la littérature des champs de lumière.

Dans un deuxième temps, (**chapitre 2**) nous nous intéressons davantage aux techniques d'apprentissage profond pour la synthèse d'images qui ont pu être proposées et développées par le passé, et qui ont nourri nos travaux.

Le **chapitre 3** est notre premier chapitre de contributions. Dans ce chapitre, nous décrivons une méthode capable de générer une gamme de nouvelles vues d'une scène, en partant d'une simple image non-annotée. Les vues sont générées latéralement autour de l'image d'entrée. Partant de cette image, notre méthode est capable, en plus de la synthèse de ces nouvelles vues, d'estimer la profondeur de la scène, et de mesurer un score de confiance par pixel quant à sa propre prédiction. Pour ce faire, la méthode repose sur la combinaison avantageuse de 2 types de réseaux de neurones.

D'abord, la méthode cherche à estimer la profondeur de l'image donnée, par le biais d'un réseau convolutionnel (*Disparity-Based Predictor*). Pendant le processus d'apprentissage, la carte de profondeur ainsi estimée est utilisée comme vecteur de disparité pour transformer la vue d'entrée ; on obtient ainsi une première prédiction basée sur l'analyse géométrique de la scène. Si cette première prédiction, qui est exclusivement construite par copie des pixels existants dans l'image d'entrée, donne déjà des résultats intéressants, elle est par nature limitée : en effet, elle ne peut pas traiter efficacement les zones qui n'étaient pas présentes dans l'image de départ (*régions occultées*).

Pour dépasser cette limite, on décide d'appliquer une méthode spécifique dans ces régions occultées. Puisque l'entrée n'est rien d'autre qu'une image non-annotée, on ne connaît pas a priori les pixels qui correspondent à ces zones occultées. On construit donc une mesure de confiance, apprise par un deuxième réseau de neurones (*Confidence-Based Merger*) guidée par les incohérences de profondeur mesurées. Finalement, dans ces zones ainsi identifiées, on applique un troisième réseau de neurones, qui va se focaliser exclusivement sur ces zones pour en améliorer l'apparence.

Le système ainsi construit est appris sur des données stéréoscopiques, et retourne des résultats convaincants, sur des jeux de données complexes tels que les jeux d'images pour voiture autonome KITTI, et se compare favorablement à l'état de l'art. D'importantes limites subsistent néanmoins, notamment liées à la sémantique très contrainte de notre méthode. Nous finissons le chapitre en explorant les limites de cette méthode et dressons une perspective quant aux possibles pistes d'amélioration.

Le **chapitre 4** étend la contribution du chapitre précédent aux champs de lumière en générant des vues selon un axe vertical en plus des vues synthétisées horizontalement dans le chapitre 3. De plus, un composant adversarial est ajouté pour augmenter la qualité perceptuelle du champ de lumière ainsi créé, en particulier dans les zones d'occultation, et le processus de calcul de confiance est amélioré. La méthode ainsi mise à jour se compare favorablement à l'état de l'art sur un jeu de données de champs de lumière.

Du fait de la difficulté de trouver des jeux de données de champs de lumière ayant des caractéristiques adaptées à notre problème, on cherche également à accroître la généricité de notre méthode en générant des champs de lumière à partir de données stéréo via un module d'adaptation stéréo - champ de lumière. Là aussi, notre modèle se compare favorablement à l'état de l'art sur le plan des métriques et sur le plan visuel. On conclut là aussi en évoquant des pistes d'amélioration envisageables.

Enfin, dans le **chapitre 5**, nous relâchons légèrement nos contraintes, et travaillons cette fois pour générer un champ de lumière entier à partir d'un sous-échantillonnage de 1 à 4 vues. Du fait de l'accroissement de la quantité d'informations présente en entrée, nous décidons d'adopter une architecture récurrente et encore plus légère, s'appuyant sur les LSTMs (*Long Short-Term Memory*). Nous construisons ainsi une méthode capable de prendre pour entrée un nombre libre de vues et de générer efficacement un champ de lumière entier dans ces conditions. La méthode est ainsi capable de s'adapter à diverses configurations d'entrée pour générer au final le champ de lumière, en s'appuyant sur une représentation MPI (*Multi-Plane Images*). Pour ce faire, nous avons recours à une architecture récurrente, en associant à chaque boucle de notre récurrence un plan de profondeur à considérer. Du fait de cette modélisation, nous sommes capables d'appliquer notre méthode dans une distribution libre de plans de profondeur de la scène au moment

du test. Cette modélisation constitue une perspective intéressante, puisque la méthode peut ainsi être utilisée "à la main" pour traiter des scènes ayant des distributions de profondeur spécifiques. Nous finissons le chapitre en explorant une fois de plus les limites relatives à l'approche présentée.

Dans ce document, nous présentons ainsi des méthodes performantes pour la synthèse de nouvelles vues dans un cadre champ de lumière, en particulier dans un contexte où le nombre de vues d'entrée est clairement sous-optimal. Le cas monoculaire (une image d'entrée seulement) est traité dans les 2 premiers chapitres de contribution, alors que le dernier chapitre de contribution s'intéresse plus particulièrement au cas où davantage d'images sont envoyées en entrée (4 ou moins). A travers les méthodes ainsi présentées, nous construisons des méthodologies et présentons des pistes qui pourront faire l'objet d'explorations ultérieures pour traiter ces problèmes avec encore davantage d'efficacité à l'avenir.

Acknowledgements

I would like to sincerely thank all the people that have accompanied me throughout these three challenging years. First, I would like to thank Christine, my supervisor, for her presence and guidance throughout the thesis.

I was very lucky to work with bright and lovely co-workers. I have no words to describe how grateful I am for their constant support, kindness and great advice.

In particular, I would like to thank Mira and Pierre for being such great friends all along the way. I would like to thank Navid for his constant support, guidance, and for his great Farsi teaching. The thesis would have been immensely more difficult if it weren't for his advice and positive presence, and I'm extremely thankful I was able to work and have discussions with such a great person. I would also like to thank Maja, for being such a positive role model to me, both as a researcher, but even more importantly, as a human being. Her great advice and her good spirit have been a huge help to overcome the various obstacles from these past 3 years. I also want to thank Ritta, for her positive influence on people, and for the kindness, empathy and support she has constantly displayed in the past year, as well as for making me discover the Armenian culture and the beauty of *Rittamojis*. I am sure she will be a great researcher, and I'm looking forward to attending her defense in a few years. I would also like to thank Elian for his presence and kindness all along these 3 years. Finally, I would like to thank Thomas for being a great officemate and for his constant support and guidance. The *lundi musique*, as well as the coffee breaks were extremely important for me, and have clearly helped me in the most difficult times of this thesis.

Of course, I'm not forgetting all the other people I was lucky to meet at inria: Jinglei and Xiaoran for their constant kindness, support, as well as for scientific discussions, Alex for the memes and the deep learning talks, Alban, Arthur and Sebastien for the video game nights, Ehsan for learning me keywords in Farsi, Christian and his little island, Pascal for the billiards sessions, Guillaume and Brandon for the badminton games, Aline

for the yoga sessions, Pierre for the support and guidance at various stages of the thesis, Reda, Anju and Tom for their kindness and sympathy... The list would be too long if I had to describe all the positive encounters I have made, and I am so grateful I had the opportunity to meet all these great people.

I would also like to thank my friends outside inria who have helped me so much: Ronan, Silex, Juliette, Maëva and Pierre, Julie and Florent, Eliette, Henri, and many many others. Thanks a lot to my parents of course, my brother and my sister, my cousins for being such positive presences, and so immense sources of support in these challenging times.

Introduction

Context

AR, VR and multi-view techniques have been getting increasingly relevant these past few years. In video games, movies, or theme parks, they have managed to captivate a new generation of users thanks to the immersive experience they provide. In the United States, the market valuation for Virtual Reality has significantly soared, object of a multiplication by 78 in 4 years, from around 205 million dollars in 2016 to nearly 16 billion in 2020. The predictions for the coming years only seem to confirm this trend.

Besides, the ongoing context, with the Covid-19 pandemic striking the whole world, comes with its likely to be deep transformations. Faced with stay-at-home orders, most of the global population has resorted to online activities a lot more. Most companies and institutes also had to adjust their activities to take into account this new dramatic situation. Cultural institutions have moved part of their production online, with broadcasted concerts and plays, as well as museums offering online visits to their customers. Other workplaces, such as estate agencies for instance, also had to adjust to the new reality of the pandemic, by providing with innovative new experiences, such as 360 visit of flats to replace on-site visits. Of course, most of these evolutions have been driven by the pandemic and its toll on societies; but they are likely to be long-lasting changes that will still be there long after Covid.

These developments mostly rely on two pillars: first, the access to data. Indeed, these technologies are built upon deep learning and data-driven techniques in general. Data is everywhere today, on Internet and in the social networks. Such data is then harvested by large companies, allowing them to build data-driven techniques (notably for imaging) with an extremely high level of efficiency.

Another founding block of these technologies is their ability to capture different viewpoints from the same scene, so as to be able to reconstruct it efficiently in its entirety.

This constraint is an important one, given that it restricts the possibilities of capturing this type of contents to specific and sophisticated equipment, that is not easy to acquire by individuals in 2021. This is though to be qualified, due to the widespread presence of smartphones, more and more of them having besides several photosensors. If such devices can constitute a valuable response to the precited concerns, they remain somewhat limited with the distance between the camera devices, necessarily bounded with the size of the smartphone itself. It implies it seems difficult to envision the processing of scenes from significantly different viewpoints.

Our contributions address these concerns, and are within the scope of the context we just described.

Motives and objectives

In this thesis, we decided to study in particular a certain type of multi-view data type: light fields. We believe processing this type of data is an interesting task, for they can be used both in an industrial framework (notably for companies like

Light fields, as explained further in chapter 1 of the present document, can be described as the representation of the intensity for every single ray of light passing through every point in space in a given scene. They can be captured using specific equipment, such as plenoptic cameras (as shown in figure 3) or camera arrays, which are in both cases impractical for everyday use.

The crux of the issue in this thesis is then as follows:

Can we reproduce light field features and behaviors, in a very accessible use case for everyday users?

Several constraints are born from this question. First, it means that we want to have a barrier of entry as low as possible for a random user who would be interested in using the method. For that reason, in all the contributions of this thesis, training and testing is systematically made from unannotated images. In the first two chapters, only 2 images are necessary for training, and only one during test time. In the last chapter, a free number of input images (between 1 and 4) can be used, and at least one more image is required

Figure 3 – The Lytro Illum plenoptic camera (source: *Wikipedia*)

during training. One of the consequences of it is that our methods can be easily employed in every day uses.

Another constraint we have decided to set is related to the complexity of the methods. Given that we want our methods to be as easy to use as possible, it implies we are interested in developing our methods while keeping a low complexity. In particular, we are very careful regarding the number of parameters we use in our methods. We are also interested in making our approaches as generic as possible, and in particular to make sure that our methods can be quickly re-trained on diverse datasets and semantics.

In this thesis, we envision the problem of light field reconstruction by the means of view synthesis: we want to be able to generate the light field given a subset of input images. We want to synthesize all the rest of the images from the light field. Theoretically, and in most cases, light field reconstruction is carried out from 4 input images, the four corner views. This is usually decided because from these 4 corner images, a significant majority of 3D points from the scene can be identified. From the 4 corner views, it is thus realistic to expect a very high reconstruction quality of the light field. In this thesis, we have decided to put ourselves in a situation where fewer views were considered. For the first two contributions, we work in the field of *monocular view synthesis*, i.e., we only have one input image at test time. For the third contribution, we freely have between 1 and 4 input images. When the number of accessible views is lower than 4, we lack information to accomplish a fully faithful light field reconstruction. One key question is then: how can

we make up for the missing information? We answer this question by using *deep learning techniques*.

Indeed, the key idea is that during training, our neural networks will have to learn the relevant information that will be missing at test time, so as to be able to process efficiently the test case. Of course, for the monocular case where only one single image is sent, this is even more true.

Our contributions

We address in the subsequent chapters the question asked above. As a starting point (in **chapter 1**), we focus on light fields, and we describe their features, their importance and their limitations. This enables us to understand the relevance of the methods that were implemented throughout this thesis. It also gives an idea of already existing works in research for light fields.

Then, in **chapter 2**, we present the state-of-the-art on deep learning techniques for image synthesis that were proposed and developed in the past, and which significantly guided our own work.

Chapter 3 is our first contribution chapter. In this chapter, we describe a method able to generate a range of new views from a scene, starting from a single unannotated image. Views are generated laterally around the input image. From this image, our method is also capable of estimating the depth of the scene, and to measure a pixelwise confidence score related to its own prediction. To do so, the method relies on an advantageous combination of two types of neural networks.

First, the methods aims at estimating the depth of the input image by the means of a Convolutional Neural Network (the *Disparity-Based predictor*). During training, the depth map is used for warping the input view; a first prediction built on the geometrical analysis of the scene is then obtained. If this first prediction, exclusively built by copying preexisting pixels in the input image, already gives interesting results, it is naturally limited: indeed, it cannot process efficiently occluded regions.

To go beyond this limit, we decide to apply a specific method in the occluded regions. Since we only have an unannotated image as input, it is not possible to know the pixels associated with these occluded areas. We then decide to build a confidence measure, learnt by a second neural network (the *Confidence-based Merger*), guided by the inconsistencies between measured depth estimates. Finally, in the areas identified as low-confidence, we apply a third neural network, which is going to target only these regions to improve the visual appearance of it.

The pipeline is learnt on stereo data, and outputs convincing results on complex datasets such as the KITTI dataset, and outperforms state-of-the-art methods. Some significant limits still exist, though, notably due to the very constrained semantics in the scene. We finish the chapter by exploring the limits of our method and draw a perspective as for the possible improvements.

Chapter 4 extends the contribution from the previous chapter to a complete light field reconstruction from monocular input views, by generating views alongside a vertical axis besides the horizontally synthesized views from chapter 3. Besides, an adversarial component is added to increase the perceptual quality of the light field, and the confidence measure process is improved. The updated method achieves good performance on a specific light field dataset.

Due to the difficulty of finding light field datasets with features adapted to our problem, we also seek to increase the genericity of our method by generating light fields from stereo contents, by the means of a stereo-light fields adaptation module. Our model also compares favorably to state-of-the-art methods, metric-wise and visually. We conclude by mentioning possible future directions for our work.

Finally, in **chapter 5**, we relax our constraints and we work this time to generate a full light field from a subsample of 1 to 4 input views. Due to the increase in amount of information present at the input, we decide to adopt a recurrent and even more lightweight architecture, based on LSTMs (*Long Short-Term Memory*). We thus build a method that can take as input a free number of input views (between 1 and 4) and that can efficiently generate a full light field in these conditions. The network constructs a MPI light field representation. We also design branches that can adapt to diverse input configurations to

generate at last our light field. We finish the chapter by exploring once more the limits of the presented approach.

Chapter 1

Light field Imaging

1.1 The plenoptic function

1.1.1 General plenoptic function

Light fields, conceptually introduced in the founding paper [1], can be defined as the representation of the light flowing within a scene. The light field function describes the amount of light flowing for every single ray of light for every point in space.

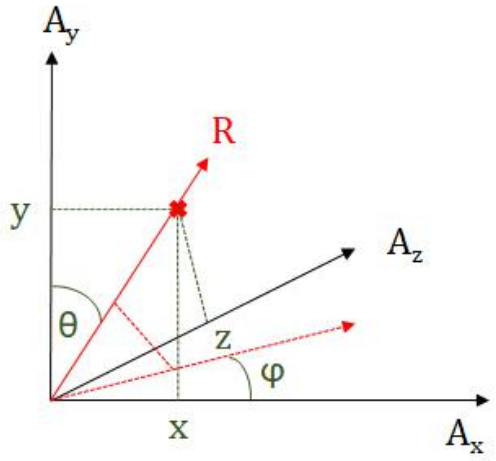

Figure 1.1 – Coordinates of a specific point for a given ray of light, used to determine input coordinates for the plenoptic function. (x, y, z) stand for the position of the considered point, we call θ the angle between the axis A_y and the ray, and ϕ the angle between the axis A_z and the projection of the ray. This parameterization is akin to spherical coordinates.

From a mathematical point of view, it is usually modelled using the *plenoptic function* P, which assigns to every point of every ray of light in a scene its current light magnitude (i.e. *the radiance* R), and can be defined as:

$$R = P(x, y, z, \theta, \phi, \lambda, t) \tag{1.1}$$

where:

— x, y, z are parameters defining the position of the considered point in space.

— θ and ϕ stand for the angles between the incident direction and orientation of the ray of light, through the usual angular measures of spherical coordinates. They discriminate the different rays of light passing through this specific point in space (see figure 1.1).

— λ stands for the wavelength and t stands for the temporal dimension, representing the evolution of the plenoptic function over time.

In this thesis, we make two assumptions:

— The sensors of capture devices subsample the wavelength, by recording light intensity in 3 color channels by using color filters.

— The considered light fields are all static, we consider no dependency in time.

Consequently, the *plenoptic function* P can be simplified as:

$$R = P(x, y, z, \theta, \phi) \tag{1.2}$$

and does not have to explicitly depend on wavelength or time in this setting.

1.1.2 2-plane representation

When capturing light fields, we particularly want to measure the plenoptic function for a given subset of rays flowing through the acquisition device. In this case, a simpler representation, first introduced in [2], can be adopted, and is shown in figure 1.2. Inside the acquisition device, every ray of light can indeed be parameterized through its intersected position with 2 defined parallel planes. Following the notations from figure 1.2, we can then define the plenoptic function as:

$$R = P(s, t, u, v) \tag{1.3}$$

It can thus be considered as a 4-parameter function, which constitutes a simpler representation for this problem. Among these 4 parameters, we thus have 2 paramaeters associated with spatial coordinates, and 2 parameters associated with angular coordinates.

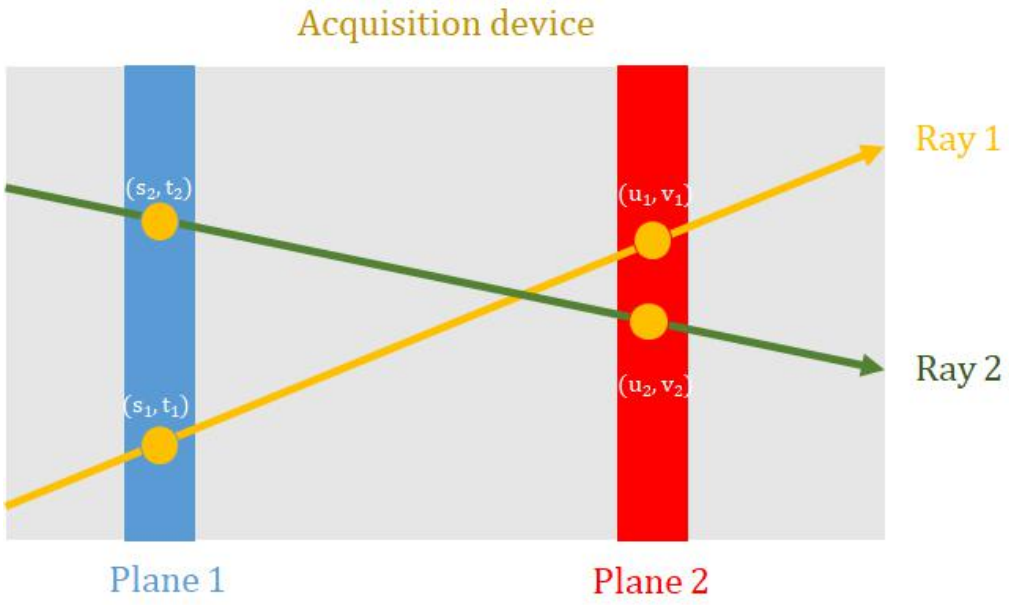

Figure 1.2 – The 2-plane representation of the light field. Inside a given volume (*acquisition device*), rays of light can be fully distinguished by 4 parameters: the spatial coordinates (s, t) and (u, v) of their intersections with two arbitrarily chosen parallel planes.

1.2 Acquisition devices

Capturing light fields is a challenging task, for it requires an ability to measure and store heavy and precise information. Plenoptic cameras are compact devices that can be constructed from classical 2D cameras by placing a micro-lens array between the main lens and the sensor.

1.2.1 Conventional photography

The process of capturing a scene using a traditional camera is described in figure 1.3. The incident rays of light are refracted by the main lens of the camera before reaching the photosensor. The final image obtained is the result of the convergence of rays onto the photosensor ; the final image I can indeed be obtained as the summation of all the rays converging to a specific position (x, y) in the photosensor.

Modifying the distance between the main lens and the photosensor allows us to focus our image onto different depth planes in the original scene. Moving the photosensor closer to the main lens indeed pushes the world focal plane further from the camera.

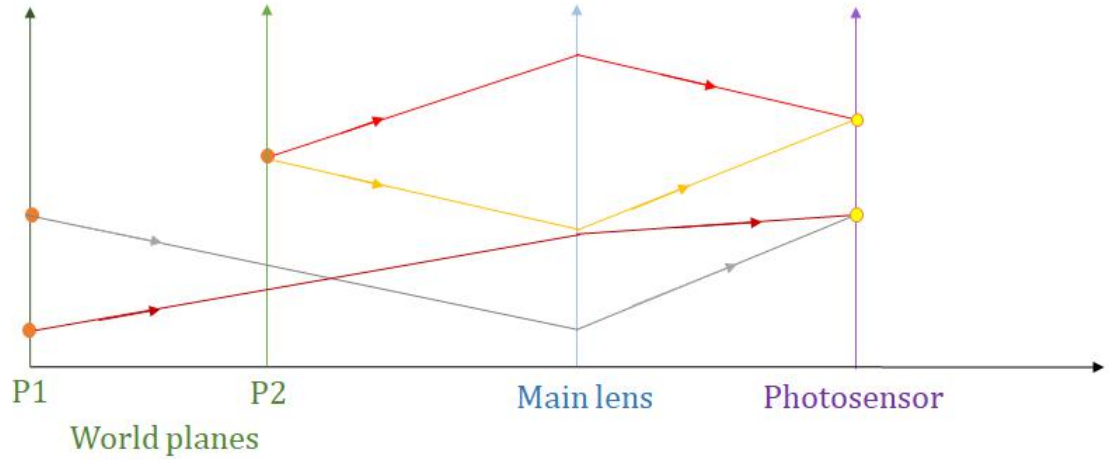

Figure 1.3 – Trajectories of the rays of light captured by a conventional camera. Points from the world planes (P2 being the focal plane) have different trajectories, and can have different starting points, but can still converge on the same pixel of the photosensor. Given that the only available information is what is arriving onto the photosensor, we cannot distinguish the contributing rays.

After capture, saved information on the photosensor is the summation, for every pixel, of the incident rays, and it is impossible to discriminate between the various contributing rays of light. Conventional cameras are therefore not suited for capturing light fields; more specific equipment is required.

1.2.2 Plenoptic cameras

Plenoptic cameras are a commonly employed equipment to capture light fields. If they are relatively recent in their current design ([4], [5]), they actually embody ideas that were developed as soon as 1908 by Lippmann. An example of such a camera (a Lytro camera) is shown on the left side of figure 1.4.

The main idea behind plenoptic cameras is to add an array of microlenses in front of the photosensor, as depicted in figure 1.5. This way, the various incident rays that would have converged onto the same point of the photosensor are consequently refracted by the microlenses, depending on their initial incidence. Thanks to this process, we have access to information related to the trajectories of the specific rays of light. Light field samples can then be captured by the camera.

Figure 1.4 – Plenoptic devices: on the left, a Lytro plenoptic camera; on the right, the Stanford camera array, presented in [3].

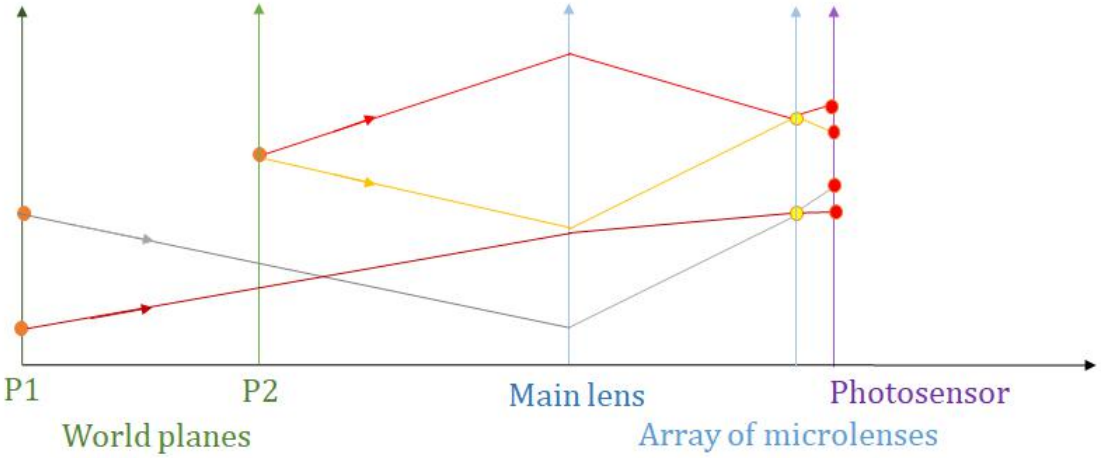

Figure 1.5 – Trajectories of the rays of light captured by a plenoptic camera. Adding an extra array of microlenses before the photosensor causes a supplementary refraction to the incident rays of light. They are refracted depending on their incidence direction, and we can then save on the photosensor information related to the origin of the rays of light.

As reminded in [6], two different designs for the optical system of the plenoptic camera have been considered and are displayed in figure 1.7: for the "plenoptic 1.0" design, or *unfocused plenoptic camera*, the focus takes place at the level of the lenslet array. Conversely, in the "plenoptic 2.0" format, or *focused plenoptic camera*, the image plane of the main lens is the object plane of the lenslet array.

Figure 1.6 – Zoom on 3 samples of a light field captured by a plenoptic camera.

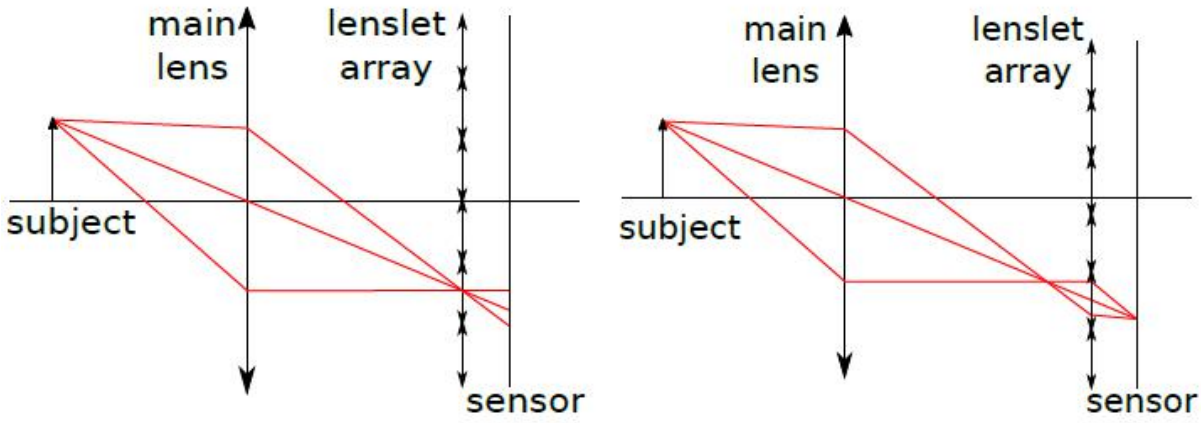

Figure 1.7 – The plenoptic camera design: on the left, the plenoptic 1.0 optical design / on the right, the plenoptic 2.0 optical design. Source of the image: [6]

When using plenoptic cameras, we usually are faced with the *spatio-angular trade-off*. The resolution of the photosensor is indeed a limiting factor; a trade-off has to be found between the resolution of the view, and the number of rays of light per pixel that is processed. It entails that a very high resolution photosensor needs to be used for plenoptic cameras, making the design impractical at times and very expensive for high-quality captures.

1.2.3 Camera arrays

Another way to capture light fields is to use camera arrays (see the right side of figure 1.4): a 2-dimensional grid of synchronized cameras, along the same plane, and with an equal distance between them (the *baseline*). The captured light field can then be repre-

sented as a 2-dimensional grid of views with a slight difference in viewpoint.

Camera arrays are challenging set-ups to build in practice, and bulky which makes them not suitable for consumer applications; plenty of difficulties arise when using them, notably related to the synchronous calibration of all the cameras simultaneously. For this reason, such set-ups are rather uncommon. Some of these camera array structures were presented in [7] and [8]. In [7], the light field is captured from a large number of cameras streaming live video, and the light field can be obtained by post-processing the significant amount of data that is obtained this way. More lightweight, the system presented in [8] can capture dynamic light fields in real time. In both cases, though, the significant baseline leads to some spatial aliasing.

More recently, commercialized cameras arrays for light fields have mostly been targeted towards 360 or virtual reality applications, such as the Google Light Field Camera. The presence of several cameras in many modern smartphones can also be assimilated to camera arrays, making light field capture potentially less cumbersome and more comfortable to use in an everyday life setting.

Other processes were also developed to produce light field contents without requiring such a bulky equipment. One possible process is to use a single conventional camera with *moving gantries*. For static scenes, the camera motion can mimic a full camera array, for an overall less cumbersome system. Light fields can also be obtained by placing a coded mask in front of the camera sensor of a conventional camera. This way, incoming rays of light are optically modulated. It was shown, notably in [9], that light fields can be efficiently reconstructed, from a compression viewpoint, by using such coded masks.

1.2.4 Different light field representations

The two types of acquisition devices capture light fields in a different way and with a different structure, as shown in figure 1.4. They are dual representations of light fields:

— On the right of figure 1.9, we can see the *sub-aperture representation*. The light field is here represented as a 2-dimensional grid of successive images of the same scene, all captured with a slightly different original position and viewpoint.

— On the left of figure 1.9, we can see the *lenslet representation*. The light field is here considered as a 2-dimensional grid of images (*microimages*), that group together

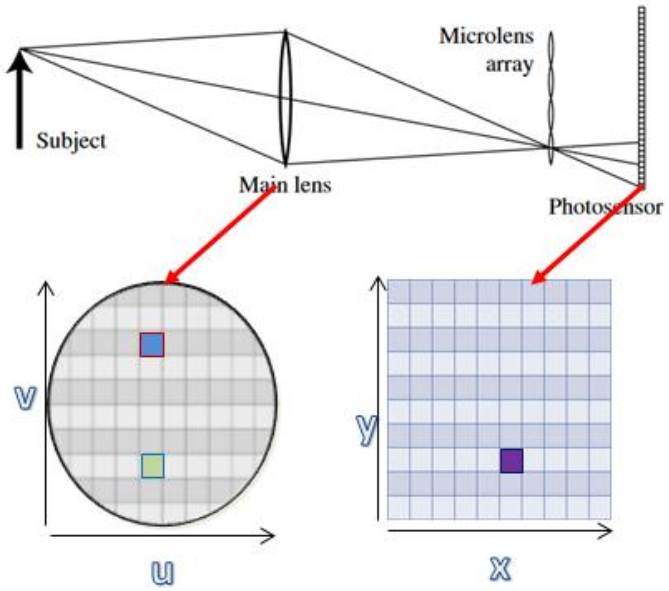

Figure 1.8 – Representation of the $x - y - u - v$ description of the light field. Source of the image: [10]

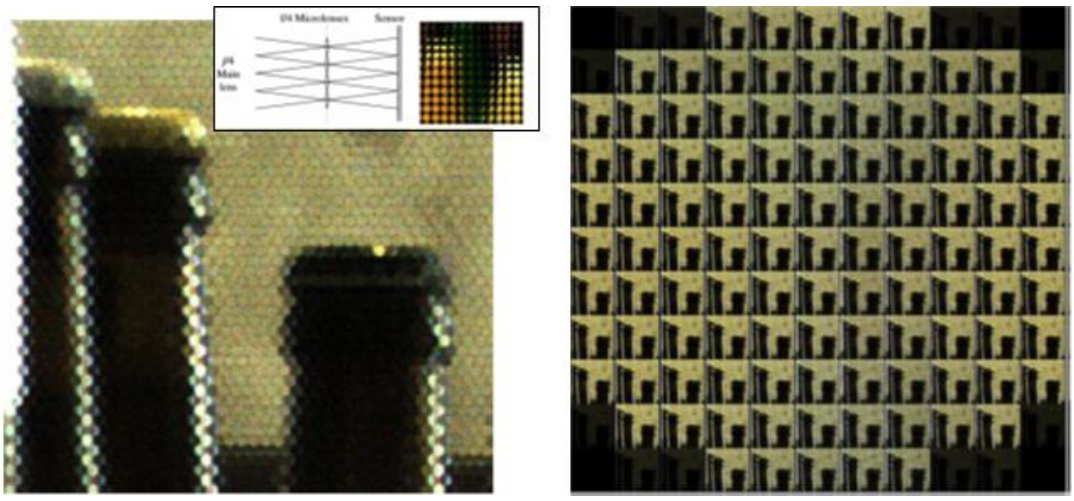

Figure 1.9 – Dual representations of the light field: on the left, the lenslet representation; on the right, the set of all subaperture images in the light field.

the same pixel position for every view. Moving from one representation to another is rather straightforward, as evidenced by figure 1.8.

Another possible representation of light fields is by the means of epipolar planes. Epipolar images are obtained by freezing an angular coordinate and a spatial coordinate; an example of such a representation is displayed in figure 1.10.

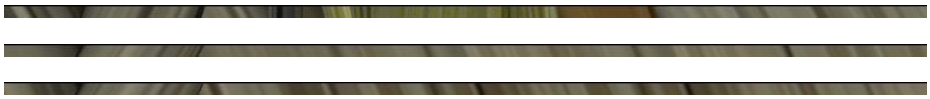

Figure 1.10 – Elements of the epipolar representation of a light field

1.3 Plenoptic applications

Thanks to the supplementary amount of information, capturing light fields brings several significant benefits when compared with conventional photography.

1.3.1 Changing viewpoints

Knowing the trajectory of the rays of light enables to simulate a virtual displacement of our camera, as shown in figure 1.9. This way, from one capture, we can predict the new views for a slight camera displacement. This application is straightforward when considering the camera array as the acquisition device, but it can also be easily achieved by recombining pixels obtained through the microlenses of a plenoptic camera.

1.3.2 Geometrical analysis and depth estimation

Light fields can be represented as a multi-view capture of the same scene. Thanks to this representation, depth prediction can be an efficient and straightforward application. Most recent light field depth estimation methods have relied on Convolutional Neural Networks to achieve this task, such as in [11], [12], [13]. Other ideas have been used to perform efficient depth estimation, such as resorting to low-rank completion, notably in [14].

The supplementary information conveyed by the plenoptic process has interesting potential developments. It naturally leads to a deeper geometrical and structural un-

derstanding of the scene. This is helpful in many applications, notably industrial ones (Raytrix,...). Besides, knowing with details the trajectory of the rays of light is helpful for other applications, notably within the framework of augmented and virtual reality.

1.3.3 Digital refocusing

In conventional photography, refocusing can be easily carried out by modifying the distance between the photosensor and the lens, shifting regions of the scene in and out of focus. For plenoptic cameras, the knowledge gained from analysing the trajectories of the various rays of light allows us to predict the hypothetically formed image for possible photosensor displacements.

Thanks to this, we can then perform *digital refocusing*, i.e. refocus the image after it was captured. Digital refocusing can be easily performed using a shift-and-add procedure on the light field, as presented in [4]. Other more efficient methods have relied on the Fourier slice theorem to perform this digital refocusing, by analyzing the light field in the Fourier domain ([15]). These methods perform digital refocusing with efficiency and good quality.

1.3.4 Other applications

We can also make advantage of the plenoptic information to tackle with more efficiency classical challenges in computer vision.

Light fields are notably convenient to improve the quality of segmentation methods, as shown in [16] and [17]. In particular, in [18], the light field structure was notably used to expand superpixels to the 4D space, in *superrays*. By exploiting the consistency between the corresponding pixels in the various views, better segmentation results are reported.

Light fields can also be used to perform more efficient inpainting. Light field recolorization and inpainting have been tackled in [19]. The work in [20] also resorts to diffusion from a structure tensor to fill in the inpainted regions of the light field efficiently.

Finally, light field-based methods have also been developed for object and face recognition, as summarized in [21]. The richer information obtained from plenoptic camera

sensors can be used in order to improve the results in this field of research, even if possibly detrimental in terms of pure complexity. Among relevant methods in this field, we can cite [22], [23], [24].

1.4 Light field processing problems

As shown in previous sections, light fields are powerful tools for improving scene analysis. Though, a significant limit of light fields is the tremendous amount of data to be stored for every capture. Besides, there is significant data redundancy between various elements of the light field. This clearly shows the interest of performing light field compression: maintaining the best possible light field features all while significantly reducing the amount of saved data. 2 examples of methods are described in the sections below.

1.4.1 Light field compression

Fourier-based compression

The redundancy of light fields in the spatial domain implies that their Fourier spectra are usually sparse. This sparsity is exploited in [25].

In this work, light fields are reconstructed from an initial randomly sampled signal in the Fourier domain. To reconstruct the full 4-dimensional signal, the frequences that will optimize, distortion-wise, the modelling of the known samples are iteratively selected. The approach is based on a pre-fixed sampling rate, and can easily be adjusted to the nature of the pre-selected samples. This method is a good and efficient way to tackle this difficult problem.

Deep learning-based compression

In [26], light field compression is carried out through the means of deep learning: an autoencoder and a 4D CNN are jointly used to recover 4D light fields from a single image, all while reducing reconstruction time.

View synthesis-based compression

Works in view synthesis can also lead to interesting developments for light field compression. Indeed, if few input images are enough to obtain a faithful reconstruction of a

light field, this is also an interesting development from a compression viewpoint.

In particular, the JPEG Pleno framework is an integral part of this field of research. The work in [27] is also an interesting work for this task. In the article, a light field compression process is shown, relying on the compression of a subset of views using HEVC inter-coding. The residual of the light field is then modeled as a video sequence, and is encoded with HEVC inter-coding, with very convincing results.

1.4.2 Light field spatio-angular super-resolution

The difficulties related to the spatio-angular tradeoff described in subsection 1.2.2 makes the problem of light field spatio-angular super-resolution extremely relevant; if we are able to generate a full light field given only a few input views, it can help us store and save light fields more efficiently. Bibliography on the domain was extensively presented in [6].

In the recent years, machine learning has been employed to perform efficient light field super-resolution. In [28], neural networks are used. First, a spatial CNN is used to restore sub-aperture images separately; then, we resort to a second CNN to generate other views. One issue with the method is the inconsistency between the separately reconstructed sub-aperture images.

One very important work in the field of view synthesis for light fields was published by Kalantari et al. in 2016 ([29]). To output high-quality images, the authors employ a cascade of two convolutional neural networks: the first CNN is used for disparity estimation, while the second CNN is efficient for color estimation. The combination of these two networks allows to process efficiently the light field. This was one of the earliest works using neural networks for producing very high-quality results in light field view synthesis.

In [30], the authors use techniques such as Principal Component Analysis and Ridge Regression to learn a matching between high and low resolution patch-volume. By exploiting the particularities of the light field structure, the method manages to return consistent results. This work was extended in [31], in which low-rank models were coupled with deep learning for superresolution.

1.4.3 Light field synthesis for view navigation

First works for multi-view synthesis were DIBR methods: the main objective of these methods was to perform view synthesis by having a prior estimation of geometry and depth for the given scene. Early works in the field include [32] (2003), [33] (2004), [34] (2008) and [35] (2010), which focus on performing multi-view synthesis through, respectively, specific texture mapping techniques and point cloud analysis. These early works led to interesting results for view synthesis, but many improvements have been observed since the publication of these methods, putting them far from the state-of-the-art today.

Among early reference DIBR methods, we can also cite the work by Chaurasia et al., published in 2013 ([36]). In this paper, an image-based rendering method was presented, based on a prior oversegmentation of the input image through superpixel analysis. The superpixels present in the various input images are connected through a graph structure, which allows to check consistencies between similar elements in various images. Besides, a depth-synthesis approach is also deployed on poorly reconstructed regions. Thanks to these ideas, the approach was able to provide an immersive navigation into an environment.

A good framework for light field reconstruction and depth prediction in light fields was also built by Wanner et al. in 2012 ([37]). In this article, the problem of stereo matching was reformulated in an attempt to take into account the specific structure of the light field, and in particular its epipolar lines. The approach was able to predict accurate depth values even for non-Lambertian regions using this particular way to model light fields. In 2014, the work was extended and improved ([38]). Here, the problem of view synthesis was framed as an inverse continuous problem, that is optimized using then state-of-the-art techniques. In particular, depth estimation is built by constraining the consistency of the epipolar lines of the depth prediction based on the consistency of the input light field. Once depth estimates are obtained, an objective function is minimized in order to optimize the quality of the produced images. If good results are obtained on most data elements, the method encounters more difficulties when processing sparse data contents. Besides, their model assumes the absence of noise or of problems in the capture, which is practically rarely true.

Another work to estimate depth was proposed in [39]. The article presents a method

that estimates depth by combining the use of defocus cues and correspondance cues from light fields. To do so, the method also exploits the epipolar structure of the light field. If the method is efficient at estimating disparities, it is not directly optimized for view synthesis, and for that reason returns suboptimal results for this task. Besides, that method being purely based on depth estimation, it is not able to process occlusions, and requires a precise prior estimation of depth.

To account for this issue, Wang et al. published in [40] a method for depth estimation that is also able to process occlusions. To identify those regions, the method analyses depth discontinuities. The method separately processes the foreground regions and background regions, improving the results of then state-of-the-art methods both in depth estimation and view synthesis.

Penner et al. published in 2017 a method for view synthesis from an unstructured set of views, called Soft3D ([41]). The method first works by estimating the geometry of the scene, and producing multiple depth maps using a stereo method. These various predictions naturally have inconsistencies for both depth estimation and occlusion processing. These inconsistencies are sorted out using a confidence from voting process. Depth uncertainty is then maintained throughout all the stages of 3D reconstruction and rendering, meaning that improvements can be performed iteratively to improve the overall result.

Methods based on spectral analysis

For light field view synthesis, a common way to proceed is to extract and exploit light field signals in specific domains, such as the Fourier domain or the shearlet domain.

Shi et al. notably exploit the sparsity of the light field continuous Fourier spectrum ([42]) to enhance the reconstruction quality of the newly produced views. Through analyses, the authors show that the sparsity of the light field is much greater in the continuous Fourier domain than in the discrete Fourier domain, due to a windowing effect. Accounting for this observation, they propose an approach to optimize for sparsity the continuous Fourier spectrum, leading to high reconstruction quality. This method for reconstruction is extremely interesting, notably for being the founding block for other methods in light field compression ([43]). Besides, its results are impressive; using this method is nevertheless not as convenient as having a set of input images set in a constant position throughout

training and testing.

The approach presented in [44] uses sparse representations of epipolar structures within light fields, by the means of adaptive shearlet transform. Using this type of representation enables the authors to efficiently reconstruct light fields.

Another example of spectral representation for light field reconstruction is presented in [45], by the resort to *Fourier Disparity Layers*. The FDL representation samples light fields according to their depth in the Fourier domain and allow efficient light field processing, for view synthesis, interpolation as well as extrapolation.

Those methods are usually built on very elegant models with impressive results. But they often also require a significant quantity of input elements for them to be efficient, that has to be chosen as relevant, unlike our methods which mostly rely on a few input natural images for training and testing.

1.4.4 Our goal

We also want to reduce the amount of light field contents that has to be saved and stored, by working on angular and spatial super-resolution. In particular:

— We will model for the rest of the thesis light fields as 2 dimensional arrays of images with slight changes in viewpoints.
— We will seek to predict all the views within the light field, given a subset of input views. In particular, the first 2 sections of our manuscript focus on the *monocular* case, i.e. reconstructing a full light field from one single image. The third chapter is devoted to light field reconstruction given several input images (between 1 and 4).

We use deep learning in our work; the next section will be devoted to descriptions of the deep learning founding blocks for view synthesis that are used throughout the thesis.

Chapter 2

Deep learning-based view synthesis

2.1 Deep learning methods

2.1.1 Fundamentals

Learning from data

Machine learning stems from the idea to build structures that can learn from data. In this context, we design models to shape the problem, with parameters to be tuned. These parameters $\mathcal{P}$ are set by checking the consistency between the predictions $\tilde{\theta}(\mathcal{P})$ of our model and the ground truth elements θ, applied on the input elements $\mathcal{I}$. We thus want to compute:

$$\arg\min_{\mathcal{P}} \mathcal{L}(\mathcal{P}, \mathcal{I}) \tag{2.1}$$

where

$$\mathcal{L}(\mathcal{P}, \mathcal{I}) = ||\tilde{\theta}(\mathcal{P}, \mathcal{I}) - \theta(\mathcal{I})|| \tag{2.2}$$

The goal is then to find the set of parameters $\mathcal{P}$ that minimizes the objective function $\mathcal{L}$. This definition implies that machine learning problems can be fully understood as optimization problems. How to tackle them fully depends on the structure of the input set $\mathcal{I}$, as well as on how we build our model $\tilde{\theta}$ and on how we define the role of the parameters $\mathcal{P}$.

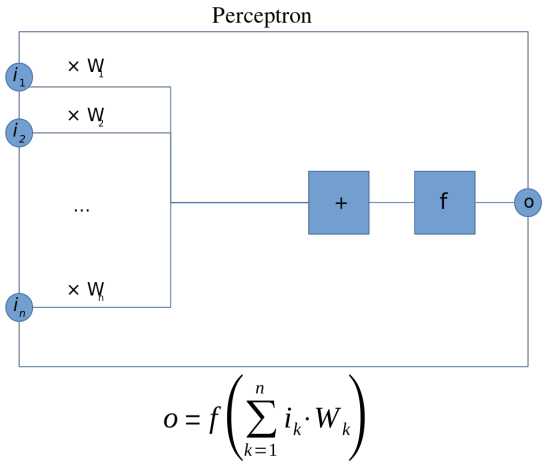

Figure 2.1 – Representation of the perceptron (source: *Wikipedia*). The prediction o is obtained by applying a nonlinear activation function to the weighted sum (by coefficients W) of the input elements i.

Structure of the model

One of the founding works in machine learning algorithm was built as soon as 1957, and is called the *perceptron* ([46]). The prediction for the model is first built as the weighted sum (by coefficients W, predicted by the method) of input elements i. A nonlinear *activation function* is then applied. A representation of it is depicted in figure 2.1.

If the perceptron has proven effective on many tasks, to tackle more complex tasks on relatively high-dimensional data, more sophisticated structures need to be designed. In the recent years, deeper and much more efficient structures such as neural networks have been used instead, which require careful optimization algorithms to efficiently learn from data.

2.1.2 Optimization and backpropagation

This section is devoted to describing the optimization algorithms that are used to minimize the metrics.

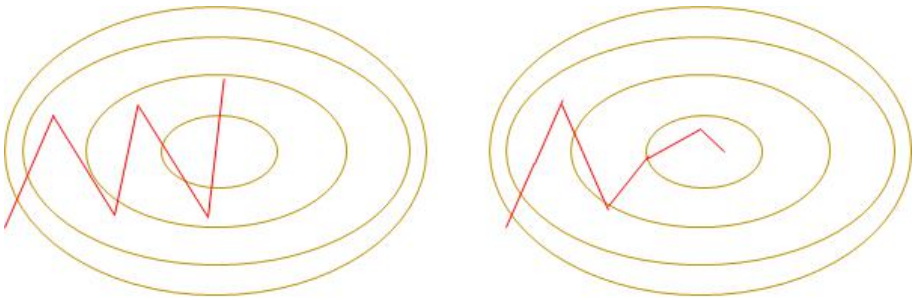

Figure 2.2 – Illustration of the impact of momentum on gradient dynamics. The yellow innermost circle represents the minimum of the surface, that we try and reach using 2 types of gradient descent methods. On the left, using standard gradient descent, the constant step size makes the function miss the minimum. On the right side, the momentum-based gradient descent is able to adjust its dynamics to previous steps, and is thus capable of converging towards the minimum.

2.1.3 Backpropagation and gradient descent

To train neural networks, *backpropagation algorithms* are used. Gradient backpropagation consists in computing the gradient of the final metrics as a chain rule of the gradients of all the layers of the network in a backward fashion until the input. Considering the problem this way allows us to use gradient descent to minimize the metrics, as long as every layer and activation function is differentiable.

Gradient descent algorithms rely on the idea to use the gradient of the target function to update the parameters at every iteration, following the equation:

$$\theta = \theta - l_r \nabla \mathcal{L}(\theta) \tag{2.3}$$

where θ represents the parameters of the problem, l_r the learning rate, and $\mathcal{L}$ stands for the optimization metrics.

2.1.4 Momentum terms

Gradient descent is a powerful algorithm, that still finds its limits when faced with unusual data structure. Indeed, on regions close to local minima similar to the one described in figure 2.2, , the "vanilla" gradient descent cannot easily find the global minimum. For that reason, a new term, the *momentum*, is usually added, which is associated with the previous value of the step size. The momentum term increases when the gradient keeps

the same direction for several successive steps, and on the contrary is reduced when the gradient changes directions. This way, it automatically reduces the step value around local minima, and increases it when it is distant from global minima. This can be mathematically expressed as:

$$\begin{cases} m_t & = \beta_1 m_{t-1} + l_r \nabla \mathcal{L}(\theta) \\ \theta_{t+1} & = \theta_t - m_t \end{cases}$$

where β_1 corresponds to the *decaying rate* and m_t is the estimation of the first moment at iteration t.

A similar expression can be used for the second momentum term, associated with the variance:

$$v_t = \beta_2 v_{t-1} + l_r \nabla^2 \mathcal{L}(\theta) \tag{2.4}$$

2.1.5 RMSprop

We would like our updates to also depend on the parameters, according to their importance. The RMSprop algorithm by Geoff Hinton (unpublished) is a good way to achieve this result. The updates are written as:

$$\begin{cases} E[\nabla^2 \mathcal{L}(\theta)]_t & = 0.9 E[\nabla^2 \mathcal{L}(\theta)]_{t-1} + 0.1 \nabla^2 \mathcal{L}(\theta) \\ \theta_{t+1} & = \theta_t - \frac{l_r}{\sqrt{E[\nabla^2 \mathcal{L}(\theta)]_t + \epsilon}} \nabla \mathcal{L}(\theta) \end{cases}$$

with ϵ being a term only used to avoid any zero situation in the numerical computation.

We can note that in this case, the update step is conditioned with the root mean squared error criterion of the gradient on the considered batch. The learning rate in RMSprop is then divided with the average of squared gradients, which are themselves decaying exponentially. This leads to significantly improved convergence.

2.1.6 The Adam algorithm

The Adam algorithm, first presented in [47], is today the most commonly optimization algorithm used in deep learning. Its main idea is to combine efficiently momentum-based convergence with ideas from RMSprop. In their case, they first update the first and second momentum terms as:

$$\begin{cases} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla\mathcal{L}(\theta) \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\nabla^2\mathcal{L}(\theta) \end{cases}$$

β_1 and β_2 are usually chosen close to 1, while m_t and v_t are usually initialized with zero values. This naturally leads to a *bias* towards the zero values for these terms. Bias-corrected terms $\hat{m}_t$ and $\hat{v}_t$ for first and second-moment estimates are then computed as:

$$\begin{cases} \hat{m}_t &= \frac{m_t}{1-\beta_1} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2} \end{cases}$$

The final parameter update can then be written as:

$$\theta_{t+1} = \theta_t - \frac{l_r}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t \tag{2.5}$$

It has been shown ([48]) that the Adam algorithm was in general the most efficient for backpropagation in deep learning, notably for image processing tasks. Following this analysis, we have decided to use the Adam algorithm to optimize the networks in all the contributions of this thesis.

2.1.7 Convolutional Neural Networks

To tackle view synthesis problems, we use neural networks, for more complex modeling, and better data exploitation. These neural networks are built on successive layers; input to the network will be transformed as it is modified through its passage through the layers, ultimately giving a prediction. When confronted with data and ground truth elements during training, the various layers and filters applied throughout the network will gradually be optimized to minimize the final metrics. Layers that are employed in the contributions of this thesis will be presented in the subsequent sections.

Convolutional layers

The key component of such networks is convolutional layers. By passing through such a layer, the input elements are spatially convolved with a filter, the parameters of which are estimated during training. Using convolutional filters entails that decisions regarding one pixel has to take into account the contributions of its spatial neighbors, which is a natural behavior in imaging; for this reason, such layers are massively employed in the field. If convolutions are powerful tools, they are nevertheless quite parameter-heavy, in particular when the input depth grows. Performing this convolution on an input element of size $W \times H \times D$ (where W stands for width, H stands for height and D for depth), using N filters of resolution $w \times h$, we have a computational cost C_{Ccost} of:

$$C_{Ccost} = DWHwhN \tag{2.6}$$

Indeed, we convolve N filters of resolution $w \times h \times D$ alongside every spatial element (of resolution $W \times H$).

Depthwise separable convolutions

Depthwise separable convolutions, first presented in [49], rely on the idea to reduce the parameter burden of standard convolutions, all while keeping their useful contributions.

The core idea is to factorize a standard convolution into 2 steps:

— first, a depthwise convolution is applied, i.e. one single filter is applied to each input depth channel.
— Then, as a second step, a pointwise convolution is applied, i.e. a 1×1 spatial convolution is applied, merging the various elements obtained from the depthwise convolution.

This implies a significant reduction in the computational cost: indeed, performing this computation on an input element of size $W \times H \times D$, using N filters of resolution $w \times h$, we have this time a computational cost C_{Scost} of:

$$C_{Scost} = DwhWH + NDWH = DWH(wh + N) \tag{2.7}$$

Leading to a computational reduction rate r_{comp}:

$$r_{comp} = \frac{C_{Scost}}{C_{Ccost}} = \frac{1}{N} + \frac{1}{wh} \tag{2.8}$$

We can this way quantify r_{comp}. We note that given that we assume N, w, and h as integers higher than 1, we always have $r_{comp} \leq 1$, and that this rate gets more significant as N, w, and h increase. Analyses and comparisons, carried out in [50], have besides shown that this reduction in computational cost does not usually lead to a significant loss in performance and efficiency when used in the right framework and architecture. Depthwise separable convolutions are thus interesting tools to reduce the computational cost all while keeping high efficiency.

Fully connected layers

Fully connected layers are layers for which every output element is obtained by weighted summation of all the input elements, and not just spatial neighbors. When applied on an input element of resolution $W \times H \times D$, using N filters, it will lead to a computational cost C_{fcCost} of:

$$C_{fcCost} = N \times H^2 \times W^2 \times D^2 \tag{2.9}$$

It allows to model connections between remote regions of the input unlike convolutional layers, but to the price of a significantly higher computational cost, given that in general $HW >> hw$. For this reason, a classical way to expand the *receptive field* (i.e. the range of input pixels that will end up connected in the graph structure) in deep image processing is to include a succession of convolutional layers instead.

Other layers

Other layers that are also commonly used in this setting include:

— *Pooling layers*: Pooling layers reduce the resolution of the input element by computing known local transformations between its elements. Max-pooling are, in particular, interesting when seeking for maximal response, while average-pooling is a convenient way to smoothen the values while downsampling.

— *Upsampling layers*: On the contrary, upsampling layers increase the resolution of the input element by computing a variety of transformations.

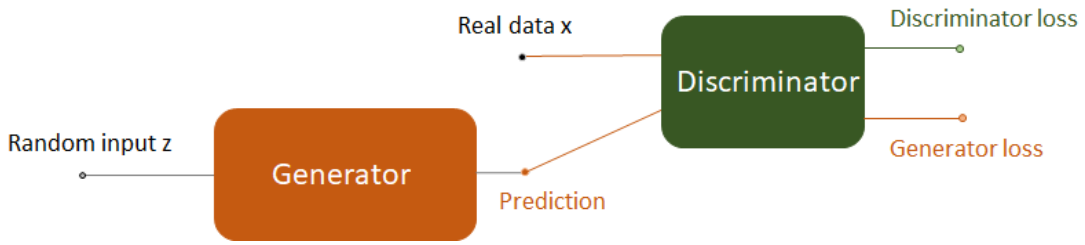

Figure 2.3 – Outline of a GAN architecture

— *Concatenating* layers can also be a useful tool to highlight information to the network, or to merge processed information.

2.2 Generative Adversarial Networks

2.2.1 Core idea

Adversarial learning is a specific branch of deep learning, introduced in 2014 by Goodfellow et al ([51]), that is particularly efficient at reproducing the same statistics as the training set. In image processing, it can notably be used to produce plausible contents and textures.

The main idea is to have two neural networks competing with each other in a zero-sum game, as described in figure 2.3:

— The *generator* outputs contents, aiming to be as realistic as possible. Its main role is to fool the other network (the *discriminator*) into believing the output contents is ground truth.
— Two propositions are then offered to the other network, the *discriminator*: the prediction by the *generator* and its ground truth equivalent. The role of the discriminator is to distinguish accurately between the ground truth and the prediction.

This indeed builds a zero-sum game: if the generator manages to fool the discriminator, it has won and the discriminator has lost. If on the other hand the discriminator was able to classify accurately, the discriminator wins. If the training process of these two methods is efficient, they are jointly trained, and can gain from the insight of the other network.

One key issue is the convergence stability of such an approach. Indeed, if the generator

is "too good" and manages to fool the discriminator every time, it has no "incentive" to improve its performance, and the discriminator is not able to distinguish and to progress, leading to a totally inefficient training process. This is also true in the reciprocal case of a "too strong" discriminator.

2.2.2 Mathematical derivation of an adversarial approach

In the original paper introducing GANs ([51]), the metrics behind such a process is described as:

$$\mathcal{L} = E_x[\log(D(x)] + E_z[\log(1 - D(G(z))] \tag{2.10}$$

D returns values between 0 and 1, that can be associated with a confidence measure as to how likely its input is to be plausible. In particular, $D(x)$ accounts for the discriminator decision when real data is sent as input, $G(z)$ is the generator prediction. As displayed in figure 2.3, x stands for the real data fed to the Discriminator, while z stands for the random input that is fed to the Generator. The adversarial problem can then be understood as an optimization problem on this metrics. The generator tries to minimize it, by guiding $D(G(z))$ towards 1, while the discriminator tries to maximize it, by pushing $D(x)$ towards 1 and $D(G(z))$ towards 0. D and G are alternatively trained to try and reach this result.

However, GANs are often faced with difficulties in their training. Notably, one key problem of GAN training is what we call *mode collapse*: if the generator is able to produce a very plausible output, the discriminator might never be able to distinguish them, leading to the absence of efficient training for both networks. In order to improve the training quality, several ideas have been developed.

2.2.3 Improving the stability of GAN training

Wasserstein loss

One way to fight mode collapse in GANs is to modify the GAN structure, for *Wasserstein GANs* (or *WGANs*) ([52]). In this case, the discriminator outputs a number assessing a confidence in the decision (not necessarily between 0 and 1). The point of the discriminator in this case is to output the highest possible number for a real element, and the

lowest possible one for a generated element. To train a WGAN, two loss functions are then jointly used:

$$\begin{cases} \mathcal{L}_d &= D(x) - D(G(z)) \\ \mathcal{L}_g &= D(G(z)) \end{cases}$$

where we keep the same notations as in the previous section, and where $\mathcal{L}_d$ stands for the discriminator metrics, and $\mathcal{L}_g$ stands for the generator metrics. The discriminator tries to maximize the discriminator metrics, in order to obtain the highest possible difference between its assessment of a generated prediction, and its assessment of a real element. At the same time, the generator tries to maximize the generator metrics, in order to try and fool the discriminator as much as possible.

Full details and analyses are shown in [52]; the Wasserstein GANs are powerful tools to improve the convergence quality of GANs.

Spectrally normalized layers

Spectral normalization for layers in Generative Adversarial Networks was first proposed in [53]. The overall idea behind the process is to normalize the spectral norm of the weight matrices.

The spectral norm of a matrix A can be written as:

$$\sigma(A) = max_{h:h\neq 0}\frac{||Ah||_2}{||h||_2} = max_{||h||_2\leq 1}||Ah||_2 \tag{2.11}$$

In other words, the spectral norm of a matrix A is the square root of the maximum eigenvalue of $A^T A$, where A^T stands for the conjugate transpose of A. The constraint we then set for every weight W of the layer is as follows: $\sigma(W) = 1$. In [53], it was shown that setting this constraint for every layer of the discriminator ensured that the Lipschitz norm of the discriminator function is bounded from above by 1, as long as the Lipschitz norm of the activation functions are also bounded by 1 from above. This was shown to be an efficient way to improve convergence of the overall adversarial approach.

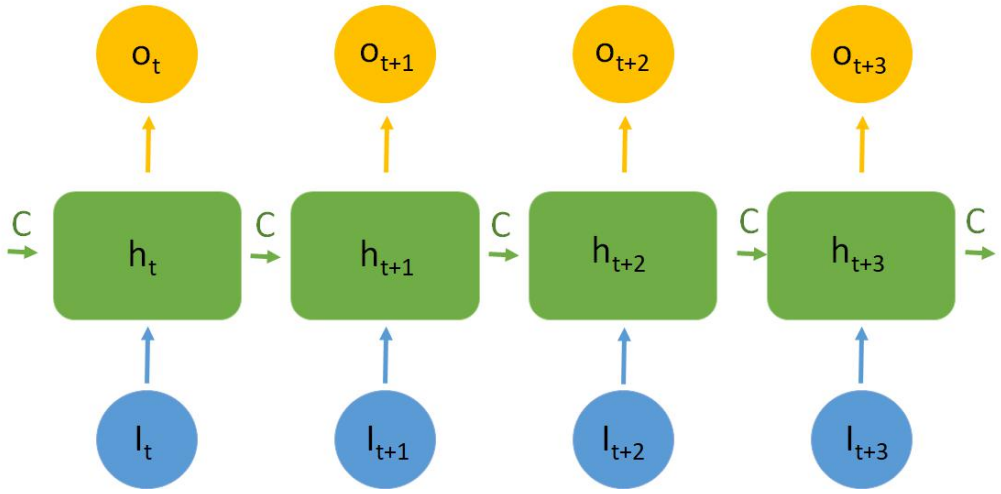

Figure 2.4 – Global outline of an unfolded Recurrent Neural Network architecture. The Recurrent Neural Network at his state h_t considers as input the element I_t of the input sequence $[I_0, ..., I_t, ...I_{N-1}]$, processes it, returns an output o_t and communicates information C to the next iteration step to update its state. The process is repeated N times (the length of the sequence).

2.3 Recurrent Neural Networks

2.3.1 Core idea

Recurrent Neural Networks (*RNNs*) constitute a specific class of neural networks that can be iteratively applied on a given sequence of inputs. The global outline of a Recurrent Neural Network architecture is described in figure 2.4.

RNNs are notably interesting for they allow to process efficiently sequences, no matter their length, with the same weights for every iteration (usually leading to lightweight architectures), all while taking into account teachings from every loop to update the weights of the network. With this strength, they are very commonly used in the fields of speech recognition and language processing.

The main issues for RNNs are related to their computational time: indeed, when training and testing a RNN, the same architecture needs to be applied sequentially; no parallel computation is possible in this case. This naturally leads to a higher computational time. Another problem that we face when using "standard" RNN is the *vanishing gradient* problem. When the gradient backpropagation algorithm goes through many layers, either

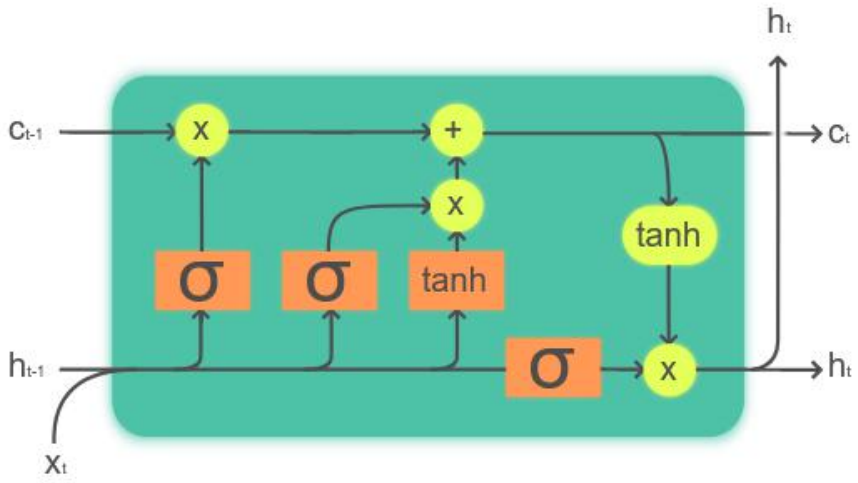

Figure 2.5 – Description of a *Long Short-Term Memory architecture*. Source: *Wikipedia*

because the neural network is very deep or either, like in this case, because the layers are passed through once per iteration, the gradient obtained by multiplying (through the chain rule) the gradients for every layer can end up being extremely small, notably if the activation functions outputs values below 1. It significantly reduces the efficiency of the learning process. To address this issue, the main idea in the literature has been to resort to LSTMs ([54]).

2.3.2 LSTM

The structure of LSTMs (*Long Short-Term Memory*) is described in figure 2.5.

LSTMs are made of a *cell*, an *input gate*, an *output gate* and a *forget gate*. For a view synthesis application, convolutional LSTMs are particularly used. The convolutional LSTM equations can be written as:

$$\begin{cases} f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + b_f) \\ i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + b_i) \\ o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + b_o) \\ \tilde{c_t} = \tanh(W_c * x_t + U_c * h_{t-1} + b_c) \\ c_t = f_t.c_{t-1} + i_t.\tilde{c_t} \\ h_t = o_t.\tanh(c_t) \end{cases}$$

where x_t is the input element to the LSTM, f_t represents the forget gate, i_t stands for

the input gate, o_t stands for the output gate, h_t is the hidden state vector, $\tilde{c_t}$ is the cell input, c_t is the cell state, W, U and b respectively stand for the weight matrices and biases learnt during training, σ_g stands for the sigmoid function, * represents a convolutional operation while . designs a Hadamard product.

For every iteration, the LSTM chooses to "forget" elements from previous iterations, and retains other useful information to update its hidden state. Such a process is a convenient way to prevent the *vanishing gradient problem*, and to make sure that long-term connections are still taken into account by the method

2.4 Classical architectures

It makes sense, when working in an image processing framework, to consider connections between spatially neighbouring regions to make decisions. For that reason, classical architectures for image processing mostly rely on a succession of convolutional layers.

2.4.1 Autoencoder structure

The autoencoder structure is a very common structure in image processing, and is displayed in figure 2.6. First, in the "encoder" branch, the input goes through a succession of convolutional layers, in which convolutional filters with a given stride are used. Passing through such an architecture, the input is processed and downsampled. Adding more convolutional layers allows to go deeper in the analysis by seeking for more global connections within the image. As for the downsampling process, it is carried out for several reasons:

— It allows to reduce the computational cost of these successive convolutional layers.
— It leads the network to compress and select the most relevant spatial data elements for the task at hand.

In the case of view synthesis, we want the output of the network to have the same resolution as the input. This is the logic behind the second branch, the "decoder", which will be tasked with upsampling the result to the original input resolution. It is usually made by combining convolutional layers with upsampling layers. At the end of these two branches, we obtain a neural network, which, when it is fed with a given input, returns a final prediction with the same resolution as the input resolution.

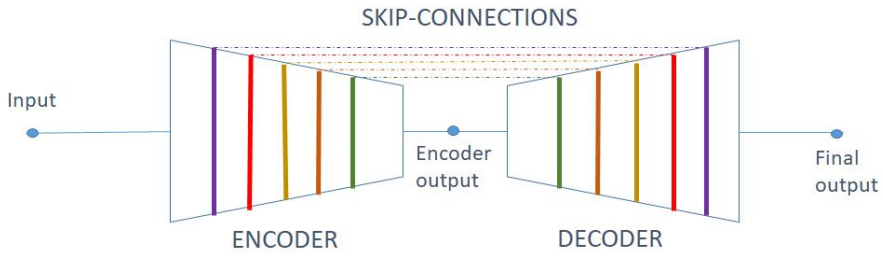

Figure 2.6 – An Autoencoder architecture with a 6-layers Encoder and a 6-layers Decoder. The input is processed with a first Neural Network (the Encoder), in which it is downsampled and convolved. Then, it is processed and upsampled with a second Neural Network (the Decoder). To make sure no relevant bit of information is lost in the whole process, we usually use Skip-Connections, i.e. we concatenate layers from the Encoder into the Decoder.

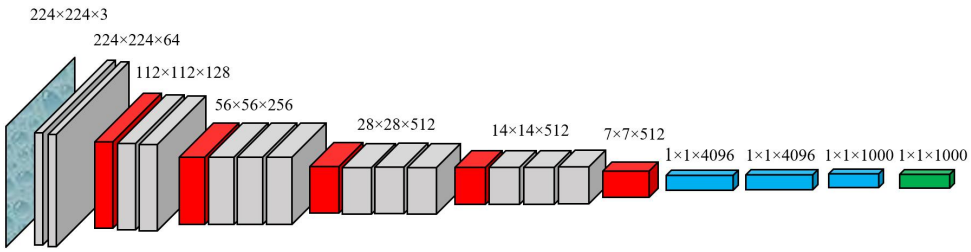

Figure 2.7 – The VGG architecture. Source: *Wikipedia*.

Nevertheless, when downsampling, then upsampling, some information is necessarily lost along the way. For that reason, we resort to *skip-connections*: layers that are part of the "encoder" branch are concatenated with layers of the "decoder" branch, as shown in figure 2.6. This way, we both have the advantages of the downsampling listed above, all while losing no bit of information in the process.

2.4.2 The VGG architecture

The VGG architecture is a canonical neural network in image classification, first proposed in [55]. Its architecture is depicted in figure 2.7. It is a very deep neural network that achieved extremely high performance on the Large Scale Visual Recognition Challenge 2014, and was widely considered as the reference neural network for image processing, replacing AlexNet ([56]) in this role.

Among its strengths compared with AlexNet, state-of-the-art at the time, we can men-

tion the resort to more layers with fewer parameters and smaller convolutional filters, as well as the resort to the ReLU activation function between every layer block.

The results obtained at the time were groundbreaking and led to its widespread use in most areas of image processing. The network was trained on ImageNet ([56]) for weeks to get this result, and is a very deep network with over 100M parameters.

2.4.3 The MobileNet architecture

MobileNet ([50]) was designed in 2017 as a response to VGG, building an alternative with equivalent performance and a much smaller number of parameters, usable in a mobile setting. To achieve this performance, the architecture relies on depthwise separable convolutions, described in section 2.1.7.

In particular, the authors suggest 4 different MobileNet architectures, depending on the value of an α hyperparameter, impacting the number of input and output channels for every layer and thus the number of parameters. In particular, they show that the approach in its largest configuration (5.2M parameters) is able to compete with VGG networks (138M parameters).

More recently, in [57], an version 2 of MobileNet was presented, modifying the structure of the various convolutional blocks to make the overall approach even more efficient with even fewer parameters.

2.5 View synthesis based on deep learning

The problem of view synthesis can be set as: how can we generate new images corresponding to new viewpoints, given access to input images from specific viewpoints? To address this issue, many different research processes have been adopted in the past.

2.5.1 Models for view synthesis

In this first section, we are going to discuss the methods, processes and deep learning layers that have been used to model problems of view synthesis in the recent years.

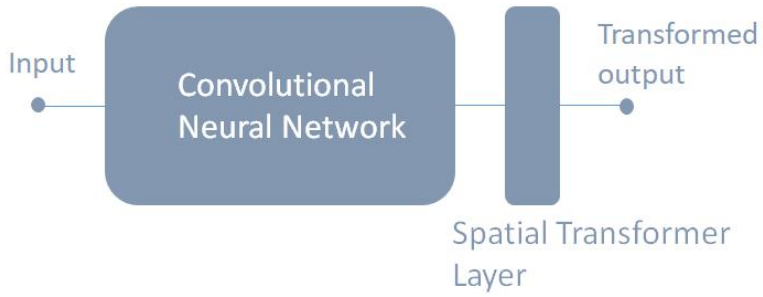

Figure 2.8 – Outline of an architecture using a Spatial Transformer Layer. This architecture allows to estimate the optimal parameters for a given spatial transformation coded in the Spatial Transformer Layer. For instance, coding the Spatial Transformer Layer as a warping process can allow to estimate disparity in an *unsupervised* way.

Spatial transformer layers

With the emergence of deep learning as a powerful tool, layers were designed to perform efficient view synthesis. Among them, a class of modules, called *spatial transformer layers*, was introduced in [58]. Spatial transformer layers are differentiable modules, usually with no extra learnable parameter, applying a specific spatial transformation on their input. The output Y of a spatial transformer layer ST when applied an input X can then be usually expressed as:

$$Y = \mathcal{T}_\theta(X) \tag{2.12}$$

where $\mathcal{T}_\theta$ is a specific spatial transformation, defined by a set of parameters θ.

These layers are a powerful transcription from DIBR methods to deep learning. Indeed, if we know the set of transformations from the input views to the target view, adopting such a structure can lead the network to predict the transformation parameters that optimize the quality of the final image produced, as shown in figure 2.8.

A specific case, that can be used in view synthesis, is to have the transformation $\mathcal{T}$ defined as a translation. In this case, the transformation-related parameters make up a disparity map or an optical flow that can be used for warping, similar to DIBR methods. The strength of such methods is to build the new views through accurate geometrical analysis, all while not having any geometrical information even during training: geometrical information is learnt in an *unsupervised* way. Furthermore, the image formation model

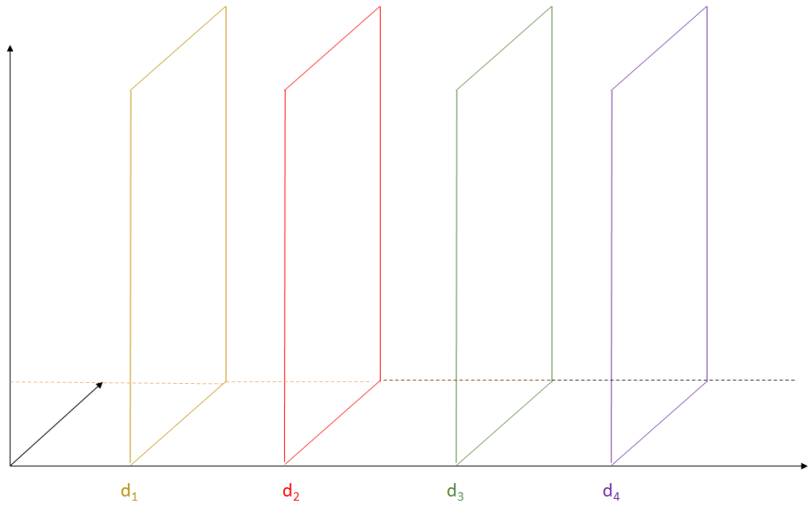

Figure 2.9 – Example of a MPI representation for a given scene. For the MPI representation, parallel planes are placed at some depth (in the figure, 4 deoth planes d_1, d_2, d_3 and d_4). We associate with each one of these planes a couple (α, C). α is the visibility map, and designates how visible the pixel at this depth level actually is. C is the colormap, giving the information regarding the color of the pixel.

is rather straightforward.

Spatial Transformer Layers are very commonly used layers in view synthesis: they have notably been used for frame interpolation in a video setting ([59], [60]), for monocular view synthesis ([61]), or depth and ego-motion learning from video ([62]).

The main issues for this class of methods are twofold: first, given that it is purely a transformation of input pixels and input views, it means that occluded regions are not explicitly processed. Besides, it also means that a small error in the transformation estimation process can lead to a strong distortion in the finally produced image. Occlusion estimation and geometrical refining are useful processes to circumvent the associated flaws, which is one of our objectives in this thesis.

Multi-Plane Imaging

Multiplanes imaging is a way to represent scenes, introduced in [63], and presented in figure 2.9. The scene is in this case modelled as a set of parallel planes at specific depths. Every plane in the representation is associated with a couple (α, C). For every plane i, α_i

is the visibility map of the pixels for this specific depth plane, ranging between 0 and 1. c_i stands for the pixel values on this specific depth plane. The final image $\hat{I}$ can then be obtained by using the *over* operator on all the D layers, as:

$$\hat{I} = \sum_{i=1}^{D} c_i \alpha_i \prod_{j=i+1}^{D} (1 - \alpha_j) \tag{2.13}$$

The disparity map $\hat{D}$ can then be obtained as:

$$\hat{D} = \sum_{i=1}^{D} \frac{\alpha_i \prod_{j=i+1}^{D} (1 - \alpha_j)}{d_i} \tag{2.14}$$

where d_i stands for the depth of plane i. The MPI model of a scene is then a collection of D depth planes (as shown in figure 2.9), each associated with a couple (α, C). This is a model that is very suitable for handling occlusions, given that pixel and visibility values for every depth plane have to be estimated. Notably, it was efficiently used for view extrapolation ([63])), light field view synthesis ([64]) and monocular view synthesis ([65]).

MPI-based predictions have nevertheless the issue of being rather complex structures, with a lot of elements to predict. When there is a scarcity of training data, it is not always easy to optimize this entire structure efficiently. And reducing the number of depth planes D to reduce the complexity of the model is not a solution, given that it forces a simpler representation of the scene, that is not necessarily accurate.

Plane-Sweep Volumes

Multi-plane imaging methods are often built upon a specific input image processing step, called *Plane Sweep Volumes* (or *PSV*). The core idea of PSVs is to provide to the network all possible disparity values for all pixels, so that it selects the best disparity associated for every pixel. Practically, it is usually performed by sending the method a sequence of input, that represent the original image shifted towards the direction of the target view with transformation vectors of increasing magnitude. The network can then build a pixelwise map for all shifted images, that can be associated with a selection mask for these pixels at this specific disparity value.

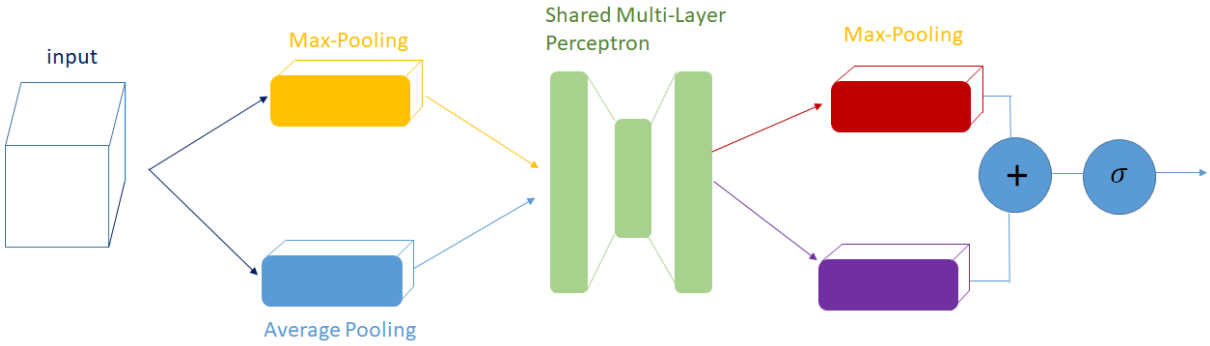

Figure 2.10 – Description of a Channel Attention Module. The input first goes through Channelwise Max-Pooling and Average Pooling. The two tensors are then fed to a Multi-Layer Perceptron. They are then added and a sigmoid function is applied. This way, the most salient features of every channel is highlighted.

Attention modules

Attention modules are CNN layers and processes that are used to lead the neural network to focus on the most relevant bit of information within the network. They are commonly employed in the literature to efficiently tackle a wide variety of tasks in computer vision, such as object detection ([66], [67]), view synthesis ([68]) or face recognition ([69]).

The attention module CBAM (*Convolutional Block-Attention Module*) employed in the thesis was first described in [70]. It is built on the combination of two specific attention modules: a channel-attention module (highlighting the most relevant elements channel-wise) and a spatial-attention module (highlighting the most relevant elements spatially across the channels). Attention modules are usually added as a residual block inside the architecture.

Channel-attention module

The Channel-attention module is described in figure 2.10. The input feature first goes through both max-pooling and average-pooling spatially to keep only one element channel-wise. Both are then passed through a Multi-Layer Perceptron, added, and go through a sigmoid activation function to obtain a Channel Attention map M_c. Multiplying the input feature with this Channel Attention tensor allows to select the most relevant and signifi-

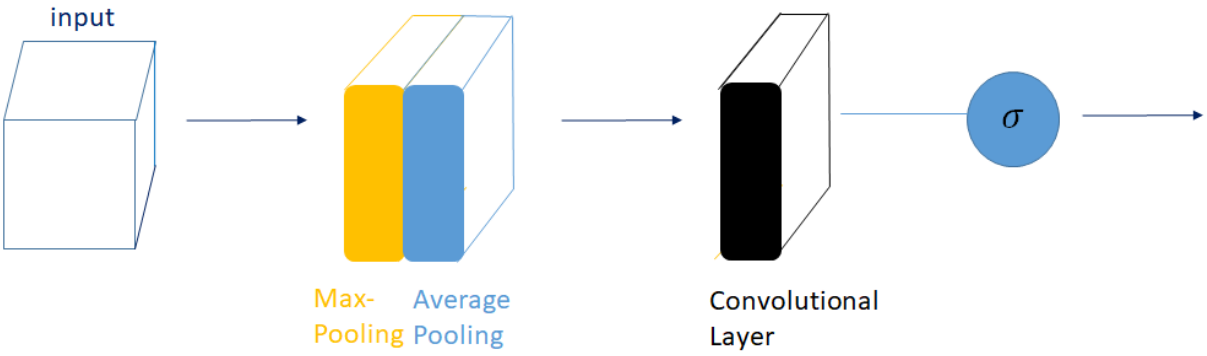

Figure 2.11 – Description of a Spatial-Attention Module. The input first goes through spatial Max-Pooling and Average Pooling; the two outputs are concatenated before being convolved. A sigmoid is finally applied. The end result is a tensor which highlights the most salient features spatially, across all channels.

cant channels for further analyses and experiments.

Spatial-attention module

The Spatial-attention module is described in figure 2.11. The input element goes through both max-pooling and average-pooling once more, but this time channel-wise to keep only one channel. After using convolutional filters, a sigmoid function is used, to give a Spatial Attention Map M_c. Multiplying the input feature with this Spatial Attention tensor allows to select the most relevant and significant spatial regions across the channels for further analyses and experiments.

2.5.2 Literature on deep learning-based view synthesis

The problem of synthesizing new viewpoints of a given scene is not new by any means. First techniques for view synthesis (*Depth Image Based Rendering* or DIBR) mostly relied on an efficient estimation of the geometry of the scene, and in particular of its depth, in order to warp new views. The efficiency of such methods is deeply dependent on the quality of the depth prediction; indeed, a small error in the depth estimation process can lead to significant distortions in the image finally produced. Besides, such an approach is not able to handle occluded regions correctly. For that reason, new approaches based on MPI and a layered description of the scene (as explained in section 2.5.1) were presented and obtained state-of-the-art results. Due to the specific structure of light fields, other classes

of methods were also developed to exploit the light field signal in various configurations (Fourier domain, shearlet domain, or in the epipolar domain).

The analysis carried out in the subsequent sections aims at describing this literature, first for view synthesis when several input views are fed to the network (*multi-view synthesis*), then when only one input image is given (*monocular view synthesis*).

View synthesis from multiple inputs

The method [64] relied on a MPI-based method. The method considers as input elements 5 images, estimates their MPI, and builds the final full light field by combining the MPI estimates, guided by the pose estimation process. The results displayed are impressive; the method is nevertheless suboptimal when working with rather small disparities. Besides, the method is highly dependent on the quality of the pose estimation process, and a small error in the pose estimation can lead to very significant errors in the end result. Besides, the method is optimized for a certain number of input views, and cannot be easily adjusted to a variable number of input views, unlike our method described in chapter 5.

In [71], the authors show a method able to reconstruct light fields by considering their epipolar structure. A CNN is trained to evaluate the quality of sheared epipolar images. The EPIs are then merged in order to reconstruct the final light field. The method returns impressive results when working for dense light fields, but its reliance on the epipolar structure naturally leads to more difficulties when the baseline increases, and in particular when distance between views becomes significant.

In [72], the authors present a new reference method for light field view synthesis, from a subset of input views. Two reconstruction modules, one based on pixels and one based on features, are used in a complementary way to reconstruct the full light field with high quality. Fusing the information obtained from applying a depth estimator to the subset of views, the approach is able to obtain significantly better results than state-of-the-art on light field datasets. If the results of this method are impressive, it is also a parameter-heavy method, which requires significant training time and parameters, as well as pretrained networks for depth estimation. Conversely, our methods are usually more lightweight, more flexible, and require a short training time, and can be trained from scratch.

The authors in [73] also address the problem of view synthesis, by first regressing from a continuous 5D representation of the scene to a volumetric representation with volume densities and view-dependent colors. This volumetric scene function is then used together with volume rendering techniques to generate novel light field views.

GANs ([51]) have also been shown to be very useful, e.g. in [74] where an unknown region is completed with a mix of pixel-wise and adversarial-based predictions, as well as for video generation [75].

View synthesis from one input

Monocular view synthesis can be defined as the aim to synthesize new viewpoints when only one image is provided as input. This problem has very interesting potential applications, but is also significantly ill-posed. Indeed, to generate those new views, the method needs to understand the geometry of the scene, as well as its semantics. Precisely, at test time, the elements present in the input image are clearly not sufficient to perform this task. As a consequence, the only way to obtain satisfying results is to define strong data priors that will be used to segment and synthesize the image finally produced.

Monocular depth estimation

Many monocular view synthesis techniques were also built on DIBR methods, meaning that we can find a clear connection between monocular depth estimation and monocular view synthesis. The first part of the section will be a non-exhaustive description of methods aiming at performing depth estimation, given only one single image at test time.

One of the early works in the field was conducted by Saxena et al. ([76]) in 2009. The results are obtained by segmenting the image into a superpixel structure. From there, they infer the 3D position and orientation of every superpixel using Markov Random Fields. This approach brought significant advancements to the research field, but is faced with the following issue: given that most decisions are carried out locally, it is very difficult to meet the objectives of global consistency that disparity maps should have.

As soon as 2014, in [77], Eigen et al. managed to show that neural networks were powerful tools for this kind of problem. In this work, an architecture was trained to learn disparity from raw pixels. The method was further improved in more recent articles ([78], [79]). More recently, in [80], depth estimation from one single image was efficiently tackled using defocus blur and semantics. These methods work in a supervised way: it is required to have access to a database of ground truth depth maps to train the network. Conversely, our method learns disparity in an *unsupervised* way; it implies that our depth estimation is likely less efficient, but on the other hand our methods can be trained on more generic datasets, without requiring ground truth depth maps.

Another whole range of methods can be described as *unsupervised* depth estimation methods: they aim at producing depth maps, but do not require any ground truth depth map for training. In those cases, two or more images of the same scene are usually sent as input, and depth is obtained from the comparison of these images. One noteworthy example of it is the work by Godard et al. ([81]), which seeks to generate two disparity maps from each image in the pair, and adds a consistency metrics to guide the final prediction. In [82], monocular depth estimation is also performed by means of a combination of supervised learning on ground truth disparity maps and unsupervised learning on pairs of images. The disparity maps obtained in both cases are very good quality, but the approaches are not themselves optimized for view synthesis and are in particular not able to process occlusions accurately. When working on datasets with significant baselines, such as KITTI, disparity gaps and thus occluded regions are significant, this can drastically change the realism of the final output. Besides, the networks in the field are usually parameter-heavy, and they are not fit for mobile applications.

View synthesis from one single image

As soon as 1997, the work in [83] dealt with this topic, using a reference method called TIP (*Tour Into the Picture*). In this method, the data priors are directly specified by the user: given an input image, the user is requested to add a fitting vanishing point for the perspective, as well as a mask specifying which pixels belong to the background, and which ones belong to the foreground. Furthermore, the user also needs to model the background and foreground scenes with a collection of simple polygons. The methods leads to good results, but it is obviously very much restricted by the need to have user input. Unlike

this one, the methods we have designed in this thesis are automatically able to process the image to perform view synthesis without requiring any human interaction.

Another work for monocular view synthesis was presented in 2005 in [84]. This method handles this time the problem in a fully automatic way. To accomplish this task, every image is assumed to be built from a single ground plane, elements sticking out and the sky. Through this representation, every pixel is going to be associated with one of the "ground", "vertical" or "sky" categories, and the horizon line can also be estimated. The classification is carried out using decision trees, by keeping labels constant in the various superpixels of the image. Thus, the algorithm can in the end return a "pop-up" version of the picture. The method is an automatic take on the challenging problem of monocular view synthesis, and returns interesting results. Still, the representation of the image is obviously fairly restricted, and is severely constrained to a certain type of image. Besides, many unpleasant artifacts are present as soon as the images slightly deviate from the chosen representation model.

The emergence of deep learning as a powerful tool for computer vision clearly was, for this specific task, a game changer. The hardest part of methods before then was the priors definition. Indeed, how is it possible to handcraft priors that will be as generic and as efficient as possible? Deep learning-based methods are in this regard attractive candidates to address the issue. Thanks to deep learning, the required priors can indeed be directly inferred from data, and do not need to be handcrafted or explicitly modelled. This way, more complex and more accurate scene representations can be learnt.

In this regard, the work by Kulkarni et al. in 2015 ([85]) was a noteworthy step forward. In this paper, the authors present what they call the DC-IGN (*Deep Convolution Inverse Graphics Network*), a model that is able to learn a specific representation of images. In particular, when given an input image of faces or 3D objects, the method is able to return a prediction of its appearance should it be perceived from a different viewpoint. To achieve such a result, the method relies on an encoder-decoder architecture. By passing the input image through the encoder, a "code group" is obtained, coding for scene latent variables, such as pose, light, texture or shape. At test time, modifying the value of the latent variables allows to generate new viewpoints from different poses or shapes. New views are efficiently synthesized using this approach and the ability to modify the

latent variables even at test time give significant interest to the approach; but it is still somewhat restricted to relatively simple datasets. Besides, the range of transformations that can be performed are mostly rotations with a slight angle.

Another related method was presented in [86]. The objective of this work is to synthesize new viewpoints of the input 3D object or input scene. In this case, the pixels are not learnt from scratch, but are copied from the input image. Notably, a 2-dimensional vector called *appearance flow* is predicted, to decide which pixels should be copied from input. The approach displays positive visual results, which were clearly state-of-the-art at the time of publishing. One of its limits is though the fact that all pixels are copied from the input image, meaning that occluded regions are not processed by the approach. Plus, the approach is very efficient at processing new viewpoints of 3D objects, but encounters more difficulties when processing natural images.

An iteration over this work was shown in [87], which is devoted to novel view synthesis, i.e. synthesizing a target image with an arbitrary camera pose, sent as a 6 degree of freedom vector, and input by the user. In particular, the model is built on the combination of a *flow prediction module*, copying existing pixels, and a *pixel generation module*, hallucinating missing pixels, by the means of a computed confidence measure. The approach is able to work from one single image, but is particularly interesting in its ability to aggregate more information and views for the analysis. Most efforts in the article are notably devoted to making this aggregation as useful and efficient as possible. Conversely, our method is specifically focused on the monocular case, and makes design choices that particularly try to take into account the scarcity of information available during training; the methods are built with a different philosophy in mind. Besides, their method returns impressive results within a specific range of transformations, present in the training data, and in particular when several images are sent as input. The method struggles when the target image is obtained thanks to a category of transformation absent from the training set. Besides, the network described in the method relies on a significant number of parameters.

In [88], the problem of novel view synthesis from a single image is once more tackled through homography estimation. Indeed, the scene is modelled in this case with a fixed number of planes. Then, homographies are predicted to model the transformation into the target view. The article presents vastly improved results when compared with previous

methods, but blur and artifacts are still present when working on natural images, notably in the stereo case. Besides, the method considers a pre-trained depth estimator for full efficiency, which implies the presence and availability of ground truth depth maps, which is uncommon when working with natural images. Besides, it also means that when changing training dataset, this depth estimator also needs to be re-trained, which drastically increases the training time. Our methods are more flexible, and can be quickly re-adjusted (with a training time ≤ 1 day) to new data if required.

Interesting works were also more specifically carried out in a stereo setting. Among them, the pioneering and reference method Deep3D ([89]) was presented in 2016. At training time, the method takes input pairs of stereo images, and produces a right-side view whenever an image is sent as input. Learning is carried out through the means of a probabilistic disparity map. The approach is efficient, but still has several drawbacks, including the inability to explicitly handle occluded regions, and its high number of parameters (about 60 million for a wide baseline and a 256×512 image), especially when the input is high resolution. Besides, the approach is not really scalable, since the size of elements of the architecture both depends on the input resolution and the target disparity range.

In [61], Cun et al. also deployed a method for monocular view generation, relying on a pre-trained depth estimator, allowing to obtain good results. Yet, the occlusions are not handled explicitly in the approach, leading to the conclusion that on sparse datasets, the method encounters difficulties; in particular, the presented method mostly focuses on multi-view dense datasets, in which disparities remain small. On the other hand, the methods described in this thesis are capable of explicitly process occlusions.

In [90], the authors deploy a method for light field generation from one single view, using a 2-stage learning process: by estimating geometry first, and then proceeding with occluded rays estimation. To build its prediction, the method learns the epipolar constraints on light fields to be able to replicate them. The method is an extremely interesting take on the subject, but remains limited with simple settings with very strong similarities (such as flower images). Besides, the resort to respecting the epipolar constraint implies that the method is clearly not optimized for high-disparity or stereo contents: in particular, the method mostly relies on the aggregation and check for consistency between the various input views during training. The architecture then requires a significant number

of input views and information during training; conversely, our method only requires 2 input images during training to output significant results.

The authors in [91] also utilize an appearance flow and spatio-angular consistent loss functions and show that their model can produce novel views of good quality in the case of densely sampled light fields as those captured by Lytro Illum cameras.

We can also cite the approach in [92] which converts a RGB-D input image into a layered depth image (LDI) representation with explicit pixel connectivity. The authors then use a learning-based inpainting model to synthesize content in the occluded regions. While the addressed problem has some similarities with the work presented in this thesis, in particular concerning occlusion handling, the authors assume a RGB-D input image while we consider a simple RGB input image.

The problem of light field view synthesis from one single image has also been recently addressed in [65] where the authors first construct a MPI representation from the input image, and then warp this MPI to generate new light field viewpoints. While this approach gives today state-of-the-art results, the network is quite heavy (around 47 million parameters). Besides, when the amount of input training data is scarce, the performance of the method tends to be sub optimal.

MPI-based approaches correspond to a certain problem modelling imposing the definition of a significant number of depth planes, that have their color and visibility maps defined by the network. Given we usually have access to no prior information regarding the data, the general way forward is to hypothesize a constant distance between depth planes. This is suboptimal; depending on the contents of the scene, it would be beneficial to have more depth planes for some depth ranges with more contents, and fewer depth planes in empty ranges.

To address this issue, a monocular approach ([93]) was proposed, in a work, ulterior to our own publications. It aims at estimating the full light field from one single image. To do so, it relies on a *Variable Multi-Plane Imaging*. As an input to the network, one image is sent, associated with a depth prediction. The analysis of the depth map gives insight as to which depth values are the most represented within the scene. Thanks to

this analysis, the authors are able to decide the sampling of the MPI to have a better scene representation. Furthermore, the method reconstructs the scene by combining 2 MPI-based networks, one devoted to visible pixels, and one devoted to occluded regions. The end result is impressive, and is a valuable step forward for the problem of monocular view synthesis. It nevertheless requires a pre-trained depth estimator, given that an estimated depth map of the scene is necessary for making the approach efficient.

In the monocular case, specific methods have also been used to tackle specifically occluded regions. MPI-based methods are an efficient way to tackle the problem, but other classes of methods have also been proposed.

Notably, in [94], Park et al. use a specific encoder-decoder network aiming at handling the occluded regions. This approach yields impressive results, but the occluded regions are not automatically identified by the network; they need to be given as ground truth occlusion maps during the training process. This implies that ground truth occlusion maps need to be easily accessible during training, which is uncommon when working with natural images, and restricts the number of training datasets that can actually be used.

Tulsiani et al. in [95] also predict the disoccluded pixels from the image they produce. To achieve this task, they use a DispNet architecture (of around 40M parameters) and a 2-layer-based view synthesis process so as to capture both the visible points and the occluded regions. It builds the estimation from a LDI structure, with a very convincing handling of the occlusions. Such a structure nevertheless comes with limits: the approach can only process occlusions inside the image, and not at the boundary, forcing the input image to be cropped. Besides, the splatting process to generate the new image imposes to halve the output resolution and leads to very significant artifacts in the image finally produced. Conversely, our methods are lighter in terms of parameters, able to handle boundary regions, and to keep input resolution in the final output.

Note that GANs have also been used in [96] to synthesize a light field from one single image, however the problem is posed as a problem of image super-resolution and the solution is therefore based on image super-resolution approaches.

Chapter 3

Monocular view synthesis for stereo output

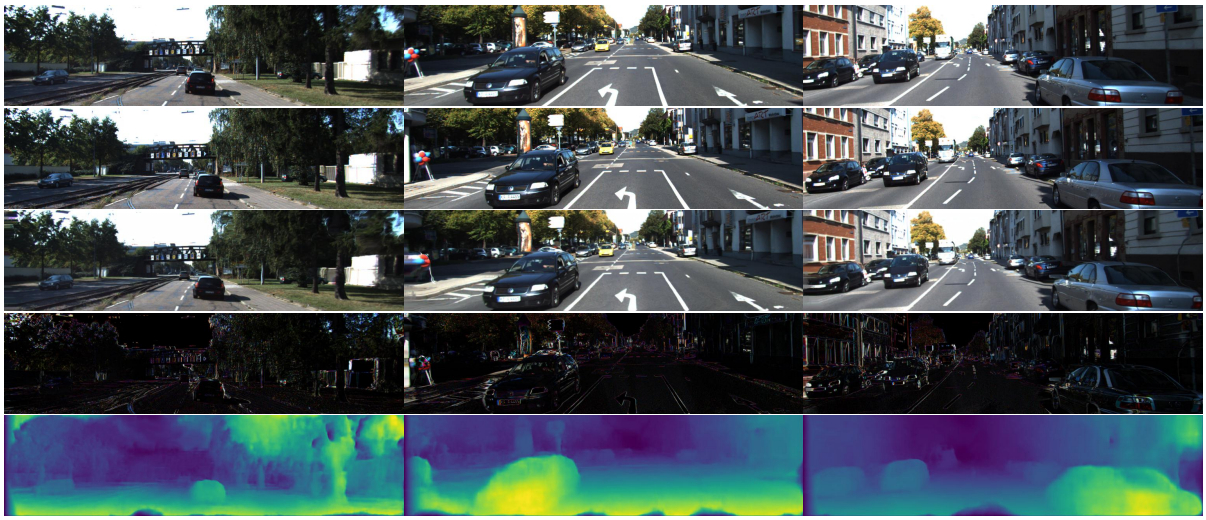

Figure 3.1 – Results of our approach on 3 examples from the KITTI test set. From top to bottom: input to the network, ground truth image, prediction carried out by the network, $l1$ error, estimated disparity map.

3.1 Introduction

Being able to synthesize new viewpoints for a given scene is a classical objective in computer vision, and it has been the subject of intense research over the past two decades. Most approaches for this task ([97], [41], [98], [99]) seek to generate those new views from multiple input frames of the scene. By contrast, synthesizing new viewpoints when given only one input image is a more challenging problem, which has garnered less attention from the vision community.

Yet, techniques aiming at generating new views from one single image can have very useful applications. First, they naturally entail a better understanding of the 3D scene from one image, which is crucial for 3D reconstruction. Besides, the past advancements and predicted developments in the coming years of multi-view formats, such as 3D, VR or light field contents, give a significant importance to these techniques. They can indeed be seen as an efficient way to compress these memory-consuming formats. Since more and more of these contents are consumed on mobile devices, it could also prove to be very beneficial for the related methods to be as computationally efficient and lightweight as possible.

Even though research on the subject is not new by any means ([83], [84]), the emergence of machine-learning based methods in the recent years has dramatically changed the prospects of the field; indeed, due to the significantly ill-posed nature of the problem, they can be especially relevant. In particular, neural networks permitted several major breakthroughs in the field of computer vision in the recent years, and they are thus tools that we prioritize for this kind of problem.

Several recent works have managed to obtain results for monocular view synthesis using deep learning techniques ([89], [61]). They often rely on a geometrical estimation of the scene from the given image, through the prediction of the pixelwise flow between the input image and the ground truth image at training time. Most of these methods are very parameter-heavy, have a hard time handling synthesis in tricky areas (occlusions, non-Lambertian surfaces,...), and capturing accurately the global structures of the image. Methods were proposed to complete the geometrical analysis with occlusion processing ([94], [88]), but most of them either are not able to process natural images yet, or require ground truth occlusion maps in the training set, which makes a generalization to more diverse data elements complicated.

In this chapter, we present a lightweight architecture able to perform view synthesis with occlusion handling in a stereo context, from one single, unlabelled and unannotated image, beyond state-of-the-art performance. Besides, it only requires a small amount of data for training. In particular, it is able, at training and at test time, to estimate the disparity map corresponding to the problem at hand, and to evaluate a confidence in its prediction when using the estimated disparity map for the synthesis. Knowing this con-

fidence measure, it is then able to refine the value of the pixels wrongly estimated, with a refinement network. The end result is a prediction built from a geometrical analysis of the scene, which is filled in wrongly predicted areas using a occlusion handling technique. Since 3D scene information is extracted in the course of the analysis, multiple new views can then be generated by interpolation. The architecture is composed of three components, a Disparity-Based Predictor (**DBP**), a Refiner (**REF**) and a Confidence-Based Merger (**CBM**).

We show the efficiency of our approach by notably applying it on the challenging, wide-baseline stereo dataset KITTI ([100], [101], [102]), with convincing and realistic-looking results for our synthesized images. We show that our method visually and metric-wise outperforms the state-of-the-art methods Deep3D ([89]) and [81] on stereo view generation, while having far fewer parameters (around 6M). We also show that it is able to perform view synthesis accurately even for scenes with large occluded regions, with no requirement to have any ground truth occlusion map for the training. Besides, it is also scalable, and can be applied efficiently on images of various resolutions. The source code as well as the trained network are publicly available.

In summary, our contributions are:

— An architecture able to outperform state-of-the-art monocular stereo view synthesis approaches, with a number of parameters reduced by an order of magnitude (5 to 10 times when compared with state-of-the-art methods in the field).

— A way of handling occlusions in a monocular setting through the learning of forward-backward consistency.

— A training schedule in 3 steps, which is key to guiding the output towards a good prediction in spite of the relatively reduced number of parameters.

— A scalable architecture, that can be applied to images of various resolutions, and that can naturally interpolate a set of high-quality views in-between the input view and the stereo predicted one.

3.2 Relevance of the problem

The starting point for this contribution was to study the field of monocular view synthesis. We decided to study this domain in particular for several reasons:

— We were interested in developing a method with a *low barrier of entry for the user at test time*. Capturing one image is very easy, in particular with the widespread presence of cameras in phones, and we wanted to design a method that could be used easily, quickly and practically by a random user from this simple experience. Monocular view synthesis was then in this framework an interesting research field.
— We were also interested in working on a problem that *had to* rely on deep learning methods as solutions. Indeed, methods based on monocular view synthesis require image understanding and segmentation from only one image, and can thus hardly be processed without resorting to data-driven methods. We were thus interested in using deep learning for a class of problems where alternatives to deep learning could clearly not compete in any way. Monocular view synthesis was then an interesting field in that regard.
— We also wanted to develop an approach that would have to draw as much information as possible from a minimal amount of available input data. Here, from one image, the method is able to return new views produced within a given range, a depth prediction, as well as a confidence in its own prediction. We felt like this task of producing all these information from one image only at test time was an interesting problem to tackle, from a learning perspective.
— Finally, we were interested in studying this problem all while having as few parameters as possible. We wanted to optimize an architecture for a tricky, ill-posed and difficult learning-based problem, that would be as lightweight as possible.

From these objectives, we drew more conclusions. First, in spite of the limitations of monocular-based methods, we wanted the method to be semantically *as generic as possible*. It is out-of-reach today to have one global network that could tackle this task with all semantics and geometric configurations, but we felt it was important to evaluate the method in real-life situations. For that reason, we trained our method on 3 different datasets and semantics. We also evaluated the network trained on automatic driving, on other datasets, as well as on natural images captured using a smartphone.

But genericity has its limits. So, we also wanted to design a method that could adjust quickly, by adapting to various, relatively small datasets during training in a short amount of time. This way, a user could fine-tune or retrain the method quickly on the new semantics and data type, using a relatively small number of input training elements. For that reason, we prioritised datasets, when possible, with several hundreds of images at

most; and we also made sure our training process only relied on pairs of images, and did not require complex ground truth elements to build, such as depth maps, or annotations.

The method described in this chapter was built to address these issues the best way possible, and the goals that were just set may help understanding design choices that were made throughout the contribution.

3.3 Preliminary research experiments

3.3.1 Image quality assessment

Once we have decided to work on monocular view synthesis, one question stands: how can we actually evaluate and assess the quality of an image? This is a tricky question, that is an ongoing research field. Following the literature and our own experiments, we decided to use 3 image metrics to evaluate our results throughout this work.

PSNR

The PSNR (*Peak Signal to Noise Ratio*) is a metrics used within the framework of view synthesis, in most works ([60], [59]). It measures the pixelwise distance between two images with the following expression:

$$PSNR = 10 \log_{10}(\frac{d^2}{MSE}) \tag{3.1}$$

where d is the color range of the pixels in the given image, and MSE stands for the mean square error.

The PSNR is a useful metrics to evaluate the faithfulness of our image compared with one ground truth element. Given that it is a pixelwise measure, it is not able to evaluate as a whole the reconstruction quality of the image and can only express a local distance between equivalent pixels. For the PSNR, the higher, the better.

SSIM

To address the concerns of a too local image metrics, we also decide to use SSIM (*Structural Similarity*) throughout this thesis (introduced in [103]). The key idea of such

an image metrics is, instead of having a pixelwise comparison, to have a measure of the structural similarity between two images. The SSIM metrics is computed on sliding windows X and Y applied on the images, and for every window, has the following expression:

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + c_1)(2\sigma_X\sigma_Y + c_2)(cov_{XY} + c_3)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)(\sigma_X\sigma_Y + c_3)} \tag{3.2}$$

where μ_X, μ_Y stand for the means of X and Y, σ_X^2, σ_Y^2 stand for their variance, and cov_{XY} their covariance, and c_1, c_2, c_3 three variables. This metrics performs a block-wise comparison between two images, which allows to have a more global assessment of image quality. For SSIM, the higher, the better.

Throughout this thesis, we have noticed that PSNR and SSIM were particularly relevant image metrics for comparing generated views of already high quality. When the image synthesis task gets more difficult, we can assume the best way to evaluate image quality is through its perceptual quality, and not through pixelwise and blockwise comparisons. For that reason, we decide to resort, in particular when we know the problem to be very difficult, to a perceptual metrics, LPIPS. For the SSIM, the higher, the better.

LPIPS

LPIPS (*Learned Perceptual Image Patch Similarity*), is a deep feature based metrics, trained to efficiently evaluate the perceptual quality of an image. It has been shown in our own analyses and experiments, as well as in works from the literature ([104]) that such a metrics was more efficient to evaluate the perceptual quality of an image, through a distance metrics to be minimized. We have noticed in particular that when working for very challenging problems where it is difficult to have a very good reconstruction quality, LPIPS was a better metrics, given it also evaluates how likely and realistic the image finally produced is. For the LPIPS, the lower, the better.

We can perform an analysis from figure 3.2, which displays one ground truth element (GT) as well as 2 predictions ($P1$ and $P2$) carried out. Naturally, one would be inclined to favour P2, because even with its flaws, it is a more realistic-looking image. Comparing PSNRs, we nevertheless notice that $PSNR(P1, GT) = 18.39 > PSNR(P2, GT) = 18.11$, while the image quality of P1 is drastically lower. We draw the conclusion that in those circumstances, the PSNR might not be the best metrics to evaluate image quality. Eval-

Figure 3.2 – An example to illustrate the relevance of using the LPIPS metrics when working on challenging problems such as monocular view synthesis. From left to right: ground truth element and respectively predictions P1 and P2.

uating those two predictions with LPIPS, we notice that we have a significantly higher LPIPS for P1 comparison than for P2 comparison: 0.326 **vs** 0.126, showing efficiently that P2 is a better prediction. We notice then that LPIPS seems to be a much more fitting metrics for evaluating image quality, in particular when the task of view synthesis is challenging.

Nevertheless, just sending the untransformed input image as prediction returns a LPIPS of 0.255, which is a better result than our P1 prediction. It implies that LPIPS does not evaluate completely efficiently the accuracy of the prediction, and clearly prioritizes the quality of the produced image. Given that we want for this challenging task to have both images as faithful as possible to the ground truth elements, and with the best reconstruction quality, we want to be able to increase, as much as possible, both PSNR (*faithfulness*) and LPIPS (*image quality*).

We have noted, though, that the disagreement between the two metrics mostly took place when the problem was really challenging, and was clearly not as pronounced when the views produced were higher quality. For that reason, throughout the thesis, we have used LPIPS (with PSNR and SSIM) as a reference metrics when working on particularly challenging problems, such as monocular view synthesis on tricky datasets, but have only relied on PSNR and SSIM for problems which had easier high-quality solutions.

3.3.2 Choosing the colorspace

Another set of experiments aimed at defining which colorspace to choose for this work. The standard choice is to use the RGB colorspace for image-related applications, but we wanted to evaluate whether other, more perceptual colorspaces such as HSV or CIELab* could prove to be more interesting.

HSV (*Hue Saturation Value*) is a colorspace decomposed into 3 different channels (HSV) than RGB. The first channel (*hue*) H codes for the color as an angular value. This allows to make sure that distance between colors are closer to actual perception than in the RGB colorspace. *Saturation* S (2nd channel) stands for color intensity, while *value* V (3rd channel) stands for color brightness (those two channels ranging from 0 to 100 value-wise). Due to the non-continuity of the *hue* channel, we instead consider in our analysis a 4-channel input with ($\cos(H)$, $\sin(H)$, S, V).

The CIE L*a*b* is a colorspace that also aims at proposing a color scale closer to human perception. It is also built from 3 channels, L*, ranging from 0 to 100 (on a black-white axis), a* representing the intensity on a green-red axis and finally b* representing the intensity on a blue-yellow axis.

Our analyses have shown that it seemed beneficial to use the RGB colorspace. First, given that most existing approaches rely on the RGB colorspace, keeping the same colorspace allows to directly re-use pre-trained weights or approaches in our framework. For this reason, the RGB colorspace already has a significant advantage over using other colorspaces.

More importantly, we significantly failed to reach equivalent performance when working with non-RGB datasets. We hypothesize several reasons for it:

— RGB data seems to be an efficient and "natural" colorspace for image representation and processing.

— Having a colorspace in which the 3 channels code for the same nature of information, and with the same range (when unnormalized) seems to be beneficial for the training process. Re-normalizing information from HSV or CIELab colorspaces to make sure they belong to the same range did not seem to help.

3.3.3 Defining the architecture of the Disparity-Based Predictor

Another important question we decided to address was related to the architecture of our Disparity-Based Predictor, which performs the first estimation by estimating a depth map. The architecture has in the end a Spatial Transformer Layer ([58]) with a translation, and we were interested in finding which architecture backbone could be the most useful for this task.

We first studied and reimplemented the work presented in [60]. The method aims at performing video frame interpolation given 2 input images. The method relies in particular on the succession of several scales within the network to address the problem. Through the aggregation of these different scales, the method is able to process the task efficiently with a relatively low number of parameters (see figure 3.3). We were interested in evaluating whether or not the architecture presented in the article could be easily extended to our problem. In this work, we show the results for two cases: the one-scale example, and the 7-scales example, where up to 7 successive scales are employed to produce the final image. A multiscale approach is in this setting equivalent to using several times our approach on gradually upsampled input.

We also evaluated the efficiency of MobileNet architectures ([50]), for various α values, given that these architectures are presented as efficient tools to perform various imaging-related tasks with a reduced number of parameters. We also checked a more recent version of MobileNet, MobileNet 2.0 ([57]), to see whether it was fit for our problem, as well as a more classical VGG16 architecture ([55]). These architectures are meant for image classification; each time, we build the architecture as an autoencoder structure, with a decoder built as the symmetrical counterpart to the encoder. We systematically train our network on the KITTI dataset ([100]) and evaluate it in this analysis on the corresponding test set. We use for every encoder architecture pre-trained weights on the ImageNet dataset ([105]). We use for all networks the same metrics, made up of 80 % of a pixelwise metrics, and of 20 % of a gradient-based metrics (difference between gradient values). We also resort to similar data augmentation for every method. What we are showing here is the metric values obtained when using such architectures, followed by a Spatial Transformer Layer, in order to warp our image prediction on the KITTI test set. We perform this analysis in the present document in order to justify the choice of our architecture. We present analyses on the test set to make sure the evaluation process is consistently carried

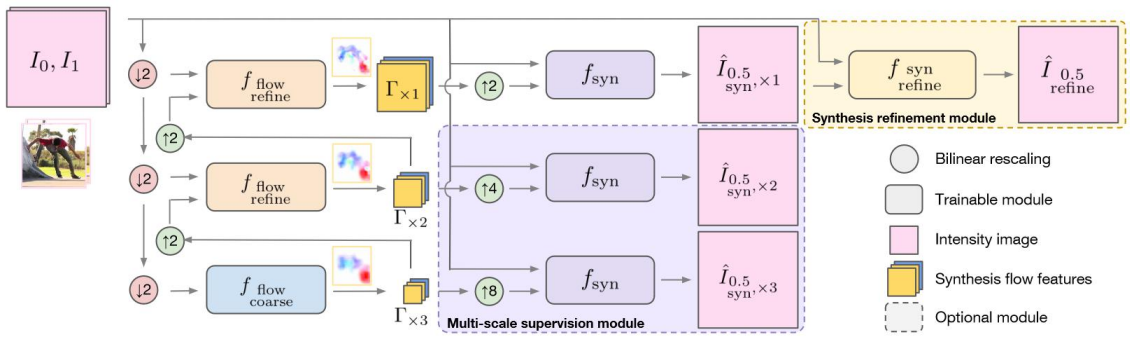

Figure 3.3 – Outline of the architecture defined in [60]

out on the same data for better comparison throughout the chapter.

The numerical results are shown in table 3.1, and some visual results are shown in figure 3.4.

We can note that metric-wise, the resort to a MobileNet 1.0 architecture seems to return the best results for our problem, with slight improvement in PSNR, and a good value of SSIM. More importantly, we can note that in LPIPS, the improvement using MobileNet 1.0 over other architectures is more significant. Besides, taking a look at the visual results, we note that MobileNet 1.0 seems to produce images with the best understanding of depth and geometry from the scene. This analysis justifies the resort to MobileNet 1.0 as the backbone for our architecture. It should be noted that we validated the architecture choice during the thesis by evaluating the performance of these various architectures on our chosen validation set. On the validation set, we had obtained a slight advantage for VGG16, but ended up choosing MobileNet 1.0 anyway due to its far more reduced number of parameters.

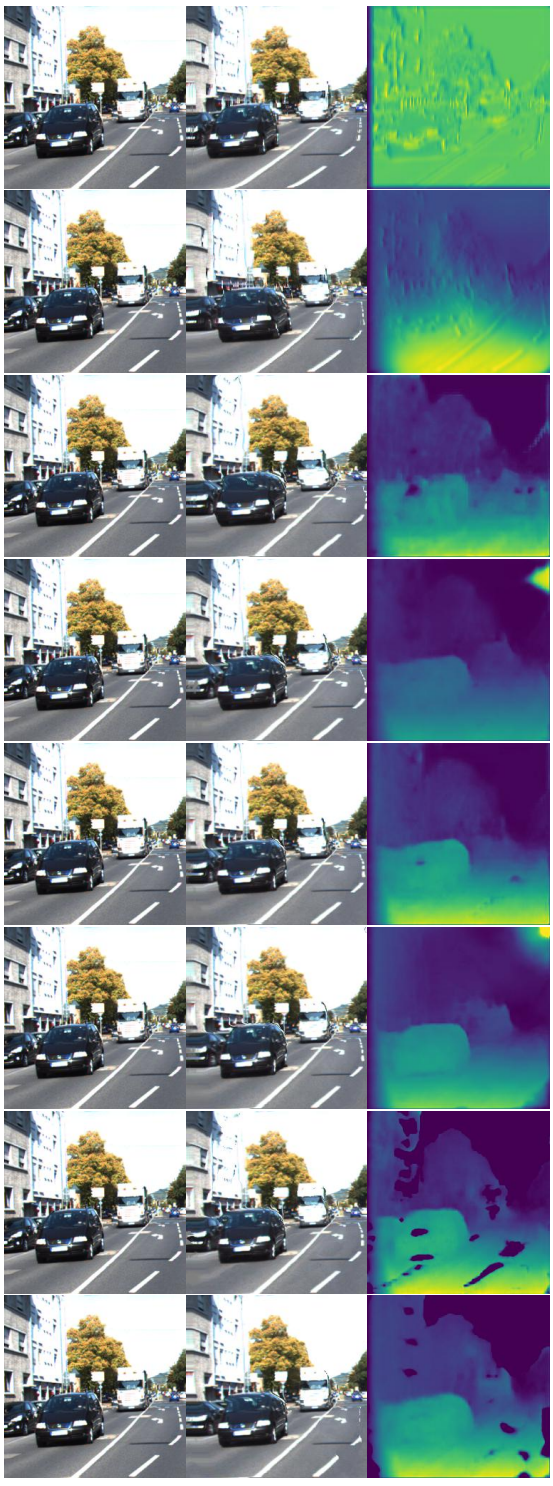

Figure 3.4 – From left to right: ground truth image, prediction and predicted depth map. For, respectively from top to bottom: [60] 1 scale, [60] 7 scales, MobileNet 0.25, MobileNet 0.5, MobileNet 0.75, MobileNet 1.0, VGG16, MobileNet 2.0. We note that MobileNet 1.0 (6th column) provides with the best depth estimation (before any further refinement) and the best visual result of all.

KITTI	PSNR	SSIM	LPIPS
[60] (1 scale)	15.70	0.64	0.217
[60] (7 scales)	16.79	0.68	0.188
MobileNet 0.25	18.00	0.71	0.162
MobileNet 0.5	18.34	**0.72**	0.154
MobileNet 0.75	18.55	**0.72**	0.148
MobileNet 1.0	**18.76**	**0.72**	**0.144**
VGG16	18.44	**0.72**	0.150
MobileNet 2.0	17.95	0.71	0.157

Table 3.1 – Numerical comparison of the values obtained on KITTI for warping from several architectures for depth estimation. We can note that MobileNet 1.0 outperforms all other chosen reference architectures for our problem. Besides, due to its lightweightedness, it seems to be the network to favour to tackle this task.

3.3.4 Using a pre-trained depth estimator?

Instead of re-training a depth estimator from scratch, we have also pondered using a pre-trained depth estimator to tackle this problem. From this idea, we have drawn the following analysis:

— We want the method to be able to adjust quickly to diverse datasets and semantics through training; in particular, we want our method to be able to work on datasets when we only have access to 2 images, and no complex ground truth element to build. For that reason, we only consider *unsupervised* depth estimators: neural networks that can estimate depth without requiring any ground truth depth element. Setting this constraint for our method allows us to use the method in a more generic way

— One constraint we have also set ourselves is to have a short training time for our method. We are in particular interested in methods that converge quickly towards the result. We also want the method to be as lightweight as possible, implying that the depth estimator also needs to respect this constraint. De facto, this means that we are not interested in using some of the state-of-the-art methods in depth estimation, given some are very parameter-heavy. Plus, they lack total generality, meaning they would need to be retrained or at least fine-tuned to be efficient in the setting we are interested in, with long training times in some cases. This adds significant constraints to our depth estimator, which are not easy to reconcile with the use of a pre-trained depth estimator.

— We have evaluated in table 3.2, for view synthesis, the positive contribution of using

KITTI	PSNR	SSIM	LPIPS
MobileNet 1.0	**18.76**	**0.72**	**0.144**
[81]	18.44	0.71	0.148

Table 3.2 – Numerical comparison of the view synthesis performance values, with prior depth estimation with [81] and without prior depth estimation, but using the view predictor based on MobileNet 1.0, both trained on KITTI. For evaluation, the method [81] was obtained from the available network provided by the authors, and fine-tuned on our own split. We can note that this evaluation process may lead to overrated results for [81], and that in spite of this possible bias, our architecture is able to outperform it.

a pre-trained depth estimator instead of using a network trained "from scratch". In other words, what we evaluate is the comparison between using our depth-based predictor when the encoder weights are pre-trained on ImageNet, and using a then-state-of-the-art approach for monocular depth estimation already trained on KITTI, [81]. We can then note that for pure view synthesis, using a "trained from scratch" architecture does not lead to a decrease in performance when compared with pre-trained depth estimators. This can be explained by the fact that the targets of the two approaches are notably different; if the pre-trained depth estimator is clearly optimized for depth estimation (which is not our primary goal), we optimize our networks for novel view synthesis. Given that the difference in performance does not seem significant, that using a MobileNet architecture requires in our case a short training time, a relatively reduced number of parameters,favouring the design of an estimator "trained from scratch" seems like a good choice.

3.3.5 Pre-trained weights on ImageNet?

Another question that may come to mind is related to the benefits of pre-training the encoder weights on ImageNet. The first benefit we gain from it is the training time to reach convergence, considerably shorter when weights are pre-trained on ImageNet (a few hours for pre-trained weigths, while the method usually takes one or two days to reach convergence otherwise). This is useful because it means that our method is able to adapt to new data elements quickly. Besides, performance-wise, as shown in figure 3.5 and in table 3.3, we have a clear benefit in pre-training these weights on ImageNet. Interestingly, we notice a particular improvement in LPIPS when pre-training the weights, showing the perceptual benefits of the process.

KITTI	PSNR	SSIM	LPIPS
Pre-trained weights on ImageNet	**18.76**	**0.72**	**0.144**
Training from scratch	18.11	0.72	0.156

Table 3.3 – Evaluating the benefits of using pre-trained encoder weights on ImageNet for our network. We notice that pre-training the encoder weights leads to a higher PSNR. More importantly, we notice we have a significant reduction in LPIPS, due to the increase in perceptual information coming with pre-training.

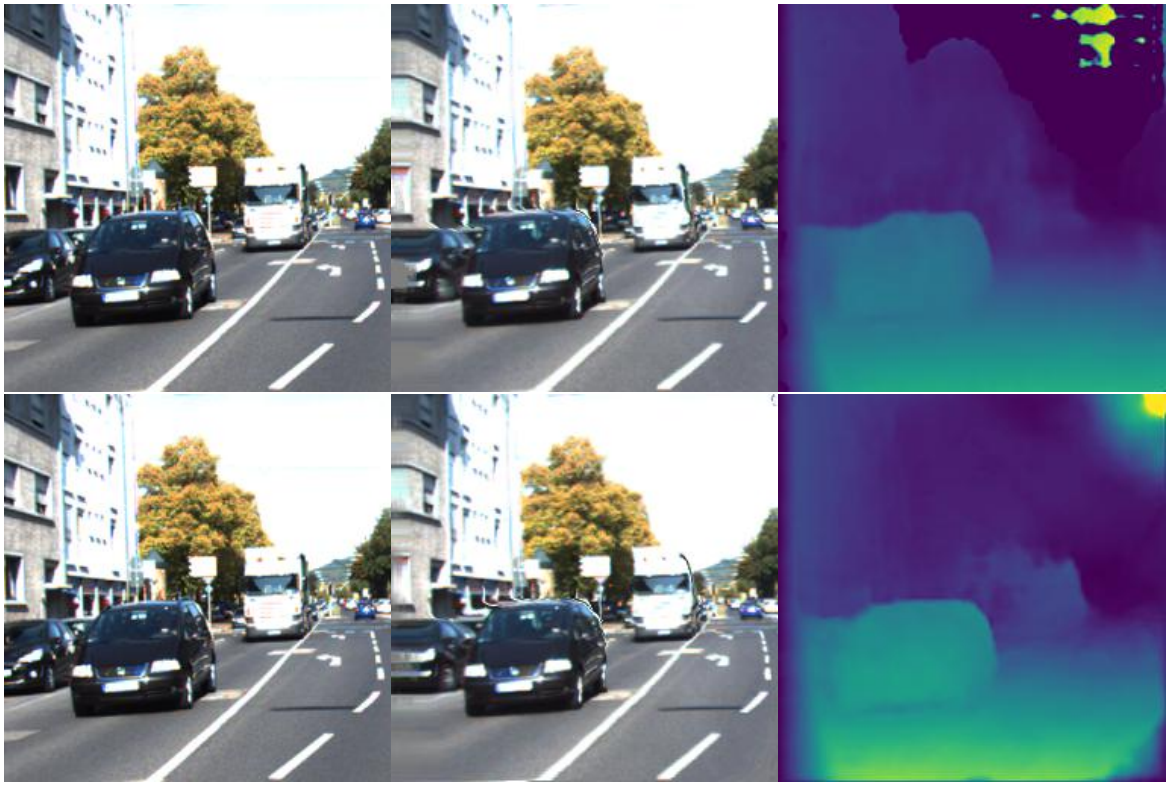

Figure 3.5 – Evaluating the relevance of pre-training the encoder weights on ImageNet (from left to right: ground truth image, prediction, estimated depth map; top row: no pre-training, bottom row: ImageNet pre-training). We notice we have a better scene representation when the weights actually go through this pre-training process.

3.4 Notations

— L, R: left and right ground truth images.
— L_{DBP}, R_{DBP}: left and right DBP-based predictions.
— L_{REF}, R_{REF}: left and right REF-based predictions.
— L^*, R^*: left and right final predictions.
— d_{LR}, d_{RL}: estimated disparity map for left-to-right (respectively right-to-left) view

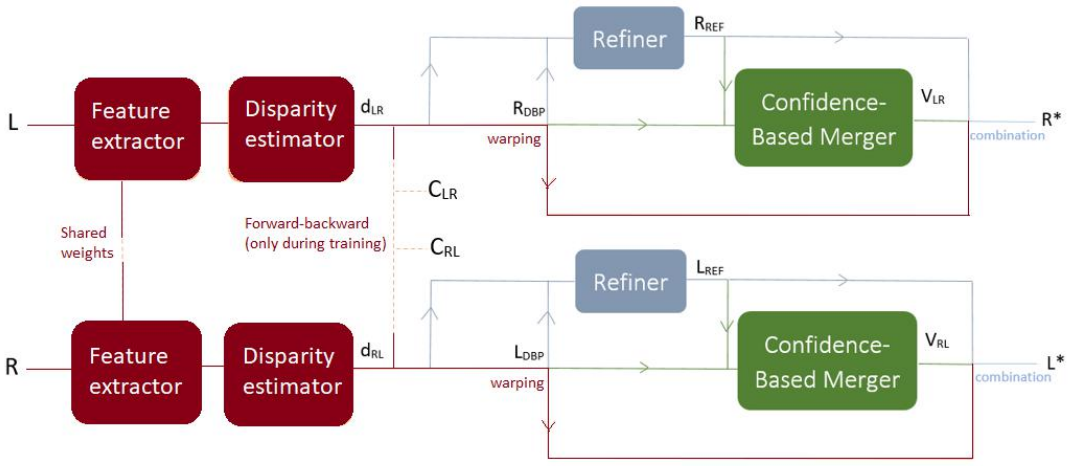

Figure 3.6 – Graph of the overall structure of our approach. Dark red blocks represent DBP, the blue block represents REF, and the green block CBM.

synthesis.

— C_{LR}, C_{RL}: confidence depth maps for left-to-right (respectively right-to-left) view synthesis as obtained by the network from the disparity estimations, only computed during training.

— V_{LR}, V_{RL}: estimated confidence depth maps for left-to-right (respectively right-to-left) view synthesis, as estimated by the network from C_{LR} and C_{RL}, so as to be used at test time.

3.5 Description of the method

In this chapter, we introduce an end-to-end differentiable approach for monocular view generation, able to synthesize new viewpoints from one single image. The work is performed in a stereo setting, meaning that the training dataset is made up of stereo pairs, with a significant disparity gap between them. Before delving into the in-depth description of every component, and into the way the learning proceeds, let us first focus on the overall structure of our approach, which is also depicted in figure 3.6. The exact architecture of every component (with number of filters, kernel sizes, strides for every single layer enumerated) is shown on `http://clim.inria.fr/research/MonocularSynthesis/monocular.html`.

3.5.1 Overall structure

The architecture can be decomposed into 3 main component networks (more details are given in their specific sections):

— The **Disparity-based Predictor (DBP)**, which seeks to estimate the disparity map between the two images at training time, by learning to warp one image from the pair onto the other viewpoint. This gives us a first prediction based on disparity map estimation. Yet, this prediction is not able to handle occlusions, and is prone to errors since the global structure of the image can hardly be captured by this pixelwise estimation.

— The **Refiner (REF)** seeks to enhance the DBP-based prediction by means of filtering. Since the main objective of this network is to cope with the flaws of the disparity-based prediction, the intuition is that it will be important for the areas that cannot be matched by disparity and for which significant information is missing at test time, such as occluded areas. The role of this network is essentially to provide a realistic and plausible output in these regions.

— The **Confidence-Based Merger (CBM)** learns the best way to combine the two complementary predictions obtained by DBP and REF, in order to obtain a good-quality final view.

Since many of these elements are actually interrelated, the learning schedule is key to guaranteeing the stability and efficiency of the approach.

3.5.2 Disparity-based Predictor (DBP)

First, we want to learn how to generate a disparity map from one given image, and then use it to predict the new view by warping the input view. We want to make sure that at test time, it can be done automatically using only one input image.

To do so, we consider a convolutional neural network, in which the last layer is a spatial transformer layer ([58]).

Learning architecture

The learning architecture is built as an encoder-decoder structure, with skip-connections so that no information is lost in the downsampling part of the encoder. The intuition behind this architecture is to consider the encoder part of the network as a feature extractor from the input, and the decoder part as a section which processes these features to generate the actual disparity map.

Following the analysis from section 3.3.3, the encoder is made of a MobileNet 1.0 architecture ([50]), where the last layers devoted to classification have been removed. The MobileNet networks are a class of lightweight neural networks which, despite their low number of parameters, are able to compete with most state-of-the-art approaches in image classification. They are characterized by the replacement of standard convolution filters within the network with a succession of depthwise convolutional filters and 1×1 pointwise convolution filters. This allows to greatly reduce the number of parameters at hand, all while maintaining a high number of feature maps. The architecture is made up of 13 successive -convolution 1×1, depthwise convolution 3×3- blocks with a gradually increasing number of filters at every block. We also initialize the weights of the encoder with pre-trained weights on ImageNet ([105]). Initializing the encoder with pre-trained weights mirrors the fact that this part of the architecture is devoted to feature extraction. As stated in section 3.3.4, we found that it brought major improvements when compared with random initialization.

We then simply design the decoder as a symmetrical counterpart to the encoder, with 5 blocks, where every block is in this case -depthwise convolution 3×3, convolution 1×1, upsampling of 2, skip-connection-. At the end of the decoder, the network returns a matrix with the same resolution as the input image, and we want this matrix to be an estimation of the disparity map of our input image. The last layer of the DBP is then a spatial transformer layer ([58]), which, similarly to [59] and [60], has no trainable parameters and uses the output of the learning architecture as a motion field to warp the input image and form the prediction. The network can then be trained directly on the images, and generate the disparity as an intermediate result.

Overall learning structure

The key idea of DBP is to use as much data as possible at training time. Since we have a pair of images accessible during training, we thus learn to perform a left-to-right, and a right-to-left view synthesis at the same time, with two independent branches (see figure 3.6). We consider that the feature extraction process is common to both tasks ; we share the weights of the encoder process in the two branches. However, we train independently the decoder (disparity estimator) in both branches. This gives us as output an access to two disparity maps d_{LR} and d_{RL} which have been independently trained from each other on their respective branches. The point of having these two disparity maps learned separately is that it will allow us to check their consistency and evaluate the confidence that we may have in our prediction for every pixel (see section 3.4). Besides, sharing the encoder weights between the two branches is also helpful since it means that the feature extractor will be fed with twice as many data elements as it would have if only one branch existed.

At test time, any of the two branches can be used individually, allowing us, depending on the chosen path, to generate a left-side or right-side view from any input image. The component, including both branches, contains around 6M parameters.

Limits of DBP

This component gives us a first prediction, based on the estimated disparity map. Now, a prediction entirely built on disparity maps has inherent flaws.

Indeed, the areas that are occluded in the input image have to be filled in the synthesized view, and the way to inpaint those pixels cannot be given by a sole disparity map. The non-Lambertian surfaces, and more globally the differences in lighting can make the matching process difficult. Besides, the performed pixelwise prediction may suffer from a loss in spatial coherence. A small error on the position of the candidate pixel can also lead to very visible artifacts.

For those reasons, we add a new component network, the **Refiner** (**REF**) which takes as input the prediction synthesized by DBP, and which has for objective to fix the issues listed above.

3.5.3 Refiner (REF)

Before we describe the architecture of REF, let us discuss the design philosophy of it.

Optimizing for a direct error metrics

We want to be able to post-process the regions where the disparity-based prediction fails. For many of these pixels, the information is actually unavailable at test time (notably in occluded areas). At this stage, we will then, in these regions, directly perform a $l1$ minimization. The network will seek to remove the artifacts and the errors produced, by refining them with neighbouring pixels. The end result runs the risk of being blurry, and of losing some important details of the scene. For that reason, we only want the Refiner to operate in areas where the DBP is not sufficiently accurate. The detection of these regions where DBP fails will be done thanks to a confidence map estimated by the third component CBM.

Architecture of REF

REF is designed as a very simple architecture made up of a succession of 8 convolutional layers, all of them (except for the last one) having 64 3×3 filters (see `http://clim.inria.fr/research/MonocularSynthesis/monocular.html` for more details).

3.5.4 Confidence-Based Merger (CBM)

The DBP and REF based predictions are both imperfect, but complementary. Indeed, DBP retains all details from the input image, but presents very strong and unpleasant artifacts when the matching is not accurate. Notably, disocclusions cannot be handled by this component. On the opposite side, REF produces an image with less artifacts, but details are lost in the process. The main objective of the CBM is to be able to combine these two predictions into one optimized final prediction. To do so, we want to be able to estimate a pixelwise confidence measure in our DBP. Indeed, if for one pixel we have a high confidence in our DBP, we will prefer the DBP pixel, since it carries more details. Conversely, if the confidence is low, the REF pixel will be preferred, with fewer visible artifacts. This will help us improve the result of our approach in occluded regions.

Confidence measure - Identification

To define this confidence measure on the DBP, we use the fact that at training time we have two estimated disparity maps: from left-to-right (d_{LR}) and right-to-left (d_{RL}) view synthesis. The following forward-backward consistency relations can be defined:

$$\begin{aligned} d_{RL}(x,y) &= d_{LR}(x - d_{RL}(x,y), y) \\ d_{LR}(x,y) &= d_{RL}(x + d_{LR}(x,y), y) \end{aligned} \tag{3.3}$$

The confidence measure is built to check whether the relations are verified for every pixel of the two disparity maps. If so, there is a consistency between the two predictions. Otherwise, it means that there was a problem in the disparity estimation process for this pixel. These relations lead us to define two confidence maps (one per branch), where γ is a parameter controlling the decay rate of the confidence measure function of the warping error :

$$\begin{aligned} C_{RL}(x,y) &= \exp(-\gamma |d_{RL}(x,y) - d_{LR}(x - d_{RL}(x,y), y)|) \\ C_{LR}(x,y) &= \exp(-\gamma |d_{LR}(x,y) - d_{RL}(x + d_{LR}(x,y), y)|) \end{aligned} \tag{3.4}$$

This way, if the relations are verified, the value for the corresponding estimated confidence will be close to 1. Conversely, if they are not, the confidence values will tend to get closer to 0.

Final synthesis - Combination

This confidence measure is available at training time, because we have access to the two images, and thus the two estimated disparities, but it cannot be used as such at test time. For that reason, a third part of the network (see figure 3.6) is devoted to learning the overall appearance of these confidence maps, from one prediction only. The architecture for learning this map is made up of 5 successive convolutional layers with (except for the last one) 32 3 $\times$ 3 filters (see `http://clim.inria.fr/research/MonocularSynthesis/monocular.html` for more details). It should be noted that we do not expect our approximation of the confidence maps to seek for the exact same values, but instead to be able to discriminate low-confidence from high-confidence pixels.

Considering the notations from section 3, the final predictions L^* and R^* can be

written as:

$$\begin{aligned} L^* &= V_{RL} L_{REF} + (1 - V_{RL}) L_{DBP} \\ R^* &= V_{LR} R_{REF} + (1 - V_{LR}) R_{DBP} \end{aligned} \tag{3.5}$$

where V_{RL} and V_{LR} are respectively the estimations of $(1 - C_{RL})$ and $(1 - C_{LR})$ carried out by CBM. Since V_{LR} and V_{RL} are initialized with values close to 0, it allows us to have as a starting point, for our final prediction, L_{DBP} and R_{DBP}. This way, we pick pixels from the disparity-based prediction when the confidence is high, and from REF when it is low. At test time, choosing either one of the two branches allows to produce a left-side or right-side view from any input image. The last activation function of the CBM is sharp, leading V_{LR} and V_{LR} to values very close to 0 or 1 ; this way we will tend to reduce the blurriness of the final result.

3.6 Learning process

Many of the components presented in the previous section are obviously interrelated, and thus a joint learning of all these components would risk to be unstable and inefficient. For that reason, a specific learning schedule needs to be specified to optimize its performance.

3.6.1 Phase I: DBP

As a starting point, we only learn the disparity-based prediction. Using the notations of section 3, we define the learning metrics as:

$$\begin{aligned} &\lambda_0(||L_{DBP} - L||_1 + ||R_{DBP} - R||_1) \\ &+ \lambda_1(||\nabla L_{DBP} - \nabla L||_1 + ||\nabla R_{DBP} - \nabla R||_1) \end{aligned} \tag{3.6}$$

We choose the l^1 metrics, following notably the analysis carried out in [106]. We jointly train the two branches, and in order to better capture the structure of the image, we add a gradient-based loss.

3.6.2 Phase II: Geometrical restructuring of DBP

To make sure that the estimated disparity map captures with as much accuracy as possible the various structures of the input image, we add a regularization step (drawing inspiration from [107]). In other words, we use the following learning metrics:

$$\begin{aligned} &\lambda_2(||\frac{2}{\max(d_{RL})}\nabla d_{RL} - \nabla L||_1 \\ &+ ||\frac{2}{\max(d_{LR})}\nabla d_{LR} - \nabla R||_1) \\ &+ \lambda_3(||L_{DBP} - L||_1 + ||R_{DBP} - R||_1) \end{aligned} \tag{3.7}$$

We constrain the normalized (to keep its value between -2 and 2) gradient of our disparity maps to be as close as possible to the gradient of our ground truth images, in order to better capture the various structures of the image. Besides, we retain a pixelwise term in the learning metrics to make sure that the prediction remains close to the ground truth element. Unlike many works ([59], [60]), we do not resort to a multi-scale approach to tackle the geometrical structuring, for the sake of reducing the number of parameters of the network.

3.6.3 Phase III: REF and CBM

We finally focus on the REF and CBM pipelines. In this last step, we freeze the weights of DBP. We do it because we do not want the whole process to interfere with the quality of the disparity maps that were produced so far. Besides, the first two steps allow to generate two disparity maps, which can then be used to generate corresponding confidence maps. By freezing the learning weights for disparity, we make sure that the confidence measure is fixed, making its estimation possible and stable.

We use the following learning metrics for our final prediction:

$$\begin{aligned} &\lambda_4(||L_{REF} - L||_1 + ||R_{REF} - R||_1) \\ &+ \lambda_5(||\nabla L_{REF} - \nabla L||_1 + ||\nabla R_{REF} - \nabla R||_1) \\ &+ \lambda_6(||L^* - L||_1 + ||R^* - R||_1) \\ &+ \lambda_7(||\nabla L^* - \nabla L||_1 + ||\nabla R^* - \nabla R||_1) \\ &+ \lambda_8(||V_{LR} - (1 - C_{LR})||_1 + ||V_{RL} - (1 - C_{RL})||_1) \end{aligned} \tag{3.8}$$

In the end, we obtain estimated inverted confidence maps (V_{LR} and V_{RL}), as well as the final predictions L^* and R^*.

3.7 Experiments

Figure 3.7 – Qualitative evaluation of the predictions carried out. From left to right: ground truth image, our prediction, L1 error between the prediction and the ground truth image.

In this section, we show the results, strengths and limits of our model. In order to evaluate its efficacy, we perform the comparison on stereo datasets with wide baselines, mostly in the context of automatic driving. The results are thus evaluated metric-wise and visually on the KITTI dataset ([100], [101], [102]). Visual results are presented in figures 3.1 and 3.7. We also advise the reader to check `http://clim.inria.fr/research/MonocularSynthesis/monocular.html`, which displays more numerous and more diverse high-resolution examples of comparisons.

3.7.1 Implementation

Before feeding them into the network, following the preprocessing steps from [59], we normalize all the images into a $[-1, 1]$ range. During training, we extract patches (with a 256×256 resolution) from the pair of images as input. We also perform color data

augmentation on-the-fly randomly for 20 % of the input elements, with random gamma and brightness transformations. Our model is trained with a batch size of 16 using Adam ([47]) as the optimization algorithm, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The network, which was implemented in TensorFlow ([108]) and Keras ([109]), has around 6.5 million parameters as a whole, and takes only a few hours to be fully trained on a Tesla P100 GPU. The learning rate is chosen as 0.0001, and is halved when there is no improvement after 10 epochs. The learning is stopped when the validation metrics has not improved after 20 epochs. When unspecified, all weights are initialized following a random normal distribution. We opt for the following values for our hyperparameters: $\gamma = 0.07$, $\lambda_0 = 0.80$, $\lambda_1 = 0.20$, $\lambda_2 = 0.85$, $\lambda_3 = 0.15$, $\lambda_4 = 0.25$, $\lambda_5 = 0.05$, $\lambda_6 = 0.50$, $\lambda_7 = 0.13$, $\lambda_8 = 0.035$.

3.7.2 Evaluation

We evaluate the quality of our method using different metrics. First, we consider PSNR as a reference measure. PSNR allows to measure the pixelwise error between the prediction and the ground truth image (the higher, the better). It is indeed a canonical measure, but it has its flaws: for one, it is not really able to measure the structural reconstruction quality of the prediction, for every pixel is considered independently from its neighbors. Besides, it can not evaluate the perceptual quality of the image produced, since a small offset of a few pixels in the prediction can drastically reduce the PSNR score, all while having a perceptual impact close to none.

To address the two precited drawbacks, we decide to use two more metrics. We resort to SSIM, since it is a metrics (the higher, the better) that is more fitting to evaluate the structural quality of the prediction, and is thus an interesting complement to PSNR evaluation. We also use LPIPS ([104]), which is a deep feature-based distance well suited for evaluating the 'perceptiveness' of our prediction (the lower, the better). Following the analysis in [104], we specifically choose the Alex-lin network for evaluation. Combining these three metrics for our prediction is a good way to have a full comparison between the various methods.

Finally, we want to evaluate the quality of our method specifically on disoccluded regions to quantify the contribution brought by our occlusion handling component. To

KITTI Test set	PSNR	SSIM	LPIPS	params
Ours	**19.24**	**0.74**	**0.139**	**6.5M**
Deep3D ([89])	19.08	**0.74**	0.220	61M
Godard et al. ([81])	18.44	0.71	0.148	30M

Table 3.4 – Statistical evaluations. The higher the PSNR and SSIM, the better. The lower the LPIPS ([104]), the better.

identify these regions on the unannotated KITTI test set, following the protocol from [95], we use an off-the-shelf stereo matching algorithm ([110]). We consider the disoccluded pixels as the pixels that could not be matched with this method. We then compute the pixelwise error only on those pixels. This allows us to evaluate the performance of the method on regions that are tricky to predict.

To evaluate our approach, we compare it with 2 reference methods: Deep3D ([89]) and Godard et al.'s approach from [81]. Deep3D aims at producing automatically a right-side image from a left-side image. Godard et al.'s work is focused on monocular disparity estimation, and we want to show that the disparity maps that we produce are better suited for warping than the output of the method that is deployed in [81].

3.7.3 Statistical results

To highlight the lightweight aspect of our network, we train our network using only the 400 pairs of frames from the training KITTI 2012 and 2015 stereo challenges. The KITTI dataset is a stereo automatic driving dataset with wide baselines. Among these 400 pairs of images, 35 are kept as validation. For evaluation, we use the 400 images from the test sets of the challenge, and perform the evaluations when working with a right-side image as input, and a left-side image to be predicted. Since Deep3D needs to be trained for a specific input resolution and disparity range, and Godard et al.'s available network is optimized for 512×256 images, we decide to center-crop the images from the KITTI test set so as to obtain 512×256 images for evaluation. The evaluation in the case of Godard et al. is the average over the pixels that were actually predicted, when using the warped disparity map predicted by the method. The statistical results are displayed in table 3.4. We can see that we obtain slightly better results in terms of PSNR and SSIM than Deep3D (and clearly above Godard et al.'s approach).

We notice that our approach significantly outperforms the other methods in terms of perceptual quality. In particular, we note that Deep3D, while close in PSNR, is very distant when looking at this perceptual metrics. We conclude that with a number of parameters very clearly lower than these state-of-the-art methods, we manage to outperform them both in PSNR/SSIM, and significantly in more perceptual measures such as LPIPS.

3.7.4 Visual comparison on KITTI

The visual difference in terms of quality of our method with the other techniques is much more significant than the numerical difference. A good way to evaluate the quality of the algorithm is thus to look at the generated images themselves, and see how realistic they are. Some results on the KITTI test set are shown in figure 3.8.

We note that visually, the images produced by Deep3D are usually more blurry and have less accurate details than in our approach. This blurring can have a pronounced effect, and it leads to the fading of certain structures (such as the sign in the second image from figure 3.8). Our approach is much sharper, and thus provides a more realistic and plausible appearance to our predicted images.

When compared with Godard et al.'s approach, we note that our method is better at handling structures and disoccluded pixels. This is particularly notable when looking at the disoccluded regions from the cars, which are not handled correctly by Godard's approach. Besides, we note that the trees in the third image are also not processed accurately by the algorithm. Generally speaking, we note that our approach is better at processing structures within the image and handling disoccluded regions.

3.7.5 Ablation study

Contribution of the various steps of the training schedule

Now, let us perform an ablation study of our approach, so as to show the benefits of using the various components of our method, and this specific training schedule. To perform these evaluations, we use the metrics defined in section 3.7.2.

The quantitative evaluations of the various components are displayed in table 3.5. Looking at PSNR and SSIM, we would be inclined to think that training the network

Figure 3.8 – Comparison of the approaches on three examples from the KITTI test set (from top to bottom: input image, ground truth image, our method, Deep3D ([89]) and Godard et al. ([81])).

Figure 3.9 – Details from KITTI views, (from left to right:) ground truth detail, detail from our prediction, detail from Deep3D, detail from Godard et al.

Figure 3.10 – Detail from views to illustrate the contribution of the Refiner. From left to right: prediction from DBP, final prediction, inverted confidence map.

end-to-end or setting no constraint on the confidence map would be the best way forward. In a way, since these two approaches are directly optimized for a pixelwise metrics, with no deviating constraint (the training schedule or the structure of the blending weights), this should come as no surprise. Yet, when taking a look at the LPIPS metrics, which accounts for the perceptual quality of the output image produced, we note that doing this actually contributes to a significant degradation of our image. We conclude from this analysis that the training schedule, defined in the article, is the best possible course of action to obtain good results from a perceptual viewpoint.

Now, taking a look at the occlusion-related metrics, we can note that understandably, adding phase II in the training schedule does not lead to a significant improvement to handle occluded regions. On the contrary, we can see that phase III brings very significant improvements in these tricky regions, which tends to validate the positive contribution of the REF.

We can now take a look at several of the images that are produced by our algorithm (see figure 3.11).

We can outline several elements:

1. Comparing columns **b** and **c**, we note at several occasions that phase II indeed improves by a significant margin the structural appearance of the produced images. This is particularly noteworthy when looking at the white truck from row 1, or the yellow building in the background from row 3.

Figure 3.11 – Elements of comparison for the ablation study. In each column, from left to right: **a)** Input image. **b)** Result from phase I. **c)** Result from phases I then II. **d)** Result from phases I, II then III. **e)** Result from phases I then III (with phase II skipped). **f)** Result when trained end-to-end using the metrics from phase III.

2. Comparing columns **c** and **d**, we note that phase III improves significantly the way occlusions are handled ; this is shown in the artifacts around the foreground car in row 1, or the artifacts from the rightmost car in row 2, which are fixed by the phase III of training. This can also be noticed when looking at figure 3.10, which zooms onto an occluded region (around the car) where artifacts are removed by the process.

3. Looking at column **e**, we note that skipping phase II usually produces results which are more blurry and less structurally sound. This is understandable by the fact that since the REF operates on a prediction which is far less accurate, it will tend to have a very strong effect to correct the flaws. We see clearly from these images that phase II is an essential component to our learning process, for by improving the quality of the intermediate prediction, it helps REF to be applied to relevant areas only.

4. Looking at column **f**, we can clearly understand the advantages of using our training schedule over an end-to-end learning process. Although in terms of PSNR and SSIM, the end-to-end output is very close to the result based on our own training

KITTI	PSNR	SSIM	LPIPS	PSNR disocc.
Phase I	18.76	0.72	0.144	14.84
Phases I-II	18.87	0.72	0.144	14.85
Phases I-II-III	19.24	0.74	0.139	15.32
Phases I-III	19.11	0.73	0.206	15.04
Phase III	19.23	0.74	0.345	15.35
No confidence	19.40	0.75	0.190	15.48

Table 3.5 – Statistical justification of the training schedule. The higher the PSNR and SSIM, the better. The lower the LPIPS ([104]), the better. PSNR disocc accounts for the PSNR on the disoccluded pixel regions only. The comparisons are carried out between networks that have been trained for the mentioned phases of the training schedule. 'Phase III' shows the case where the network is fully trained end-to-end with the metrics from phase III. 'No confidence' shows the result when, during phase III of the training schedule, the metrics constraining the blending weights to be based on the consistency of the disparity maps is removed.

schedule (see table 3.5), we can see that visually, the difference is very significant: our training schedule allows us to obtain a result which is less blurry and far more accurate. By forcing the training process to follow a certain schedule, we thus make sure that our result remains convincing from a perceptual viewpoint.

Constraining the confidence map

We now compare the results that we obtain when setting a forward-backward confidence constraint in the definition of our blending weights, and when we set no constraint. The comparisons are performed in figure 3.12.

We see that the images that we end up obtaining when we do not specify any constraint over the confidence map are usually much more blurry, which is confirmed by the significant difference in terms of LPIPS shown in table 3.5. Besides, figure 3.12 shows that confidence maps are good representations of occluded or non-Lambertian regions, and that by removing the constraint, we also lose this valuable information.

3.7.6 Interpolation process

One of the interesting features of the approach is that it is not only able to produce stereo views, but that it can also generate a sequence of good-quality interpolated views between the input image and the prediction.

Figure 3.12 – Elements of comparison for constraint set on the confidence map (yellow in confidence maps means low-confidence in DBP prediction). (from left to right: **a)** Prediction when FB-constraint is set. **b)** Corresponding confidence map. **c)** Prediction when no FB-constraint is set. **d)** Corresponding confidence map.)

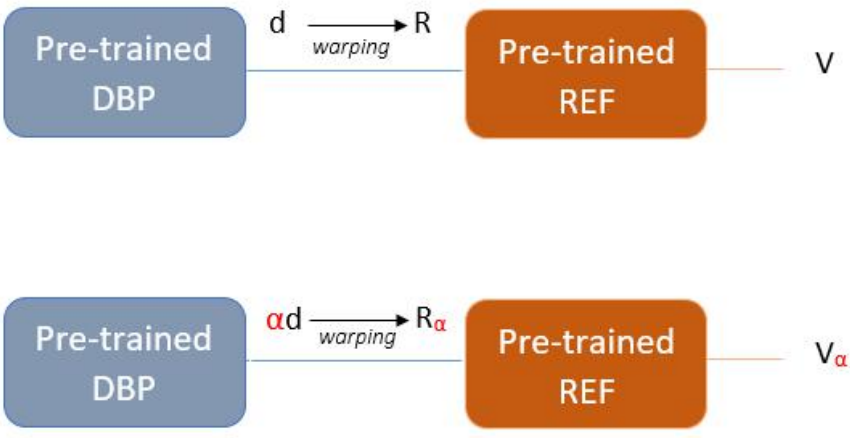

Figure 3.13 – Diagram showing the interpolation process. We scale the disparity map obtained as an output of the pre-trained DBP. Warping the input view with the scaled disparity map yields a first interpolated view R_α. The pre-trained Refiner (and the rest of the network) will then deal with the artifacts, and a confidence map will be generated, so as to produce the final interpolation V_α with good quality.

To do so, we use an already trained network. We scale the disparity map that is ob-

Figure 3.14 – Interpolation results (top row) and confidence maps.

	V0	**V1**	**V2**	**V4**	**V5**	**V6**
Ours (PSNR)	**37.05**	**39.55**	**43.39**	**43.36**	**39.45**	**36.94**
(SSIM)	**0.96**	**0.97**	**0.99**	**0.99**	**0.97**	**0.96**
[90] (PSNR)	33.74	35.65	41.04	40.87	36.58	34.25
(SSIM)	0.92	0.94	0.98	0.98	0.94	0.92

Table 3.6 – Metric-wise comparisons between our method and [23] on the Flowers test set. Vi represents the i-th view in the central line of the light field. V3 is the central view (so, the input), and is thus not evaluated. V0 and V6 are the target views, while all the other ones are obtained through our interpolation process. We note that our approach clearly outperforms the work from [90], metric-wise, on all views, even for the interpolated views.

tained as output of the DBP. We use the scaled disparity map to compute by warping a first approximation of the interpolated view. Using the rest of our pre-trained pipeline on the warped views allows us to obtain a good-quality sequence of views around the input image.

Visual results of this interpolation process are shown in figure 3.14, as well as the corresponding confidence maps built for every interpolated image, when working with large baseline stereo sets, such as KITTI. Visual examples are also shown in the supplementary video on: `http://clim.inria.fr/research/MonocularSynthesis/monocular.html`.

In addition, we evaluate quantitatively this interpolation process by working on light field content (the smaller-baseline Flowers dataset, introduced in [90]). We train our network on the Flowers training set, by considering stereo pairs (either 'leftmost view - central view' or 'central view' - 'rightmost view') on the central line of the light field. We then evaluate our approach on all the views from this central line, by performing our interpolation process. This way, we are able to evaluate visually and metric-wise the quality of the images that we produce when the central view of the light field is used as input. We compare our results with [90], a method generating a full 4D light field from one single

image, using the code provided by the authors. The comparisons are only performed on the central line, and are displayed in table 3.6. We note that our approach clearly outperforms [90] on these interpolated views. This shows that our interpolation process is efficient in producing good-quality interpolated views. Besides, it also shows that our approach is able to work efficiently on various datasets, with various baselines and semantics.

3.7.7 Results on other datasets

The network has been trained on the KITTI training set and evaluated on the KITTI test set, but it can be applied efficiently on any kind of images from urban scenes when trained on KITTI. This is shown in figure 3.15.

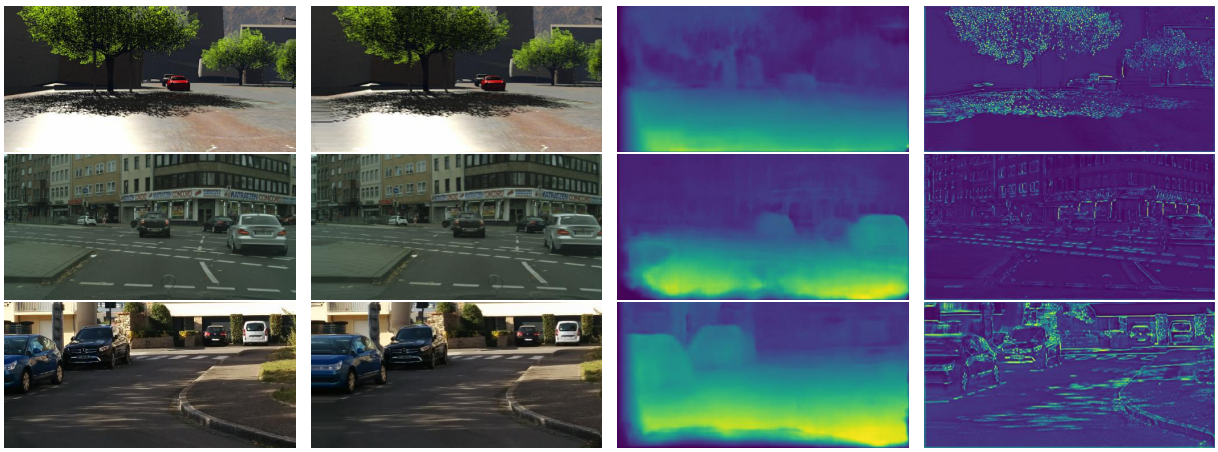

Figure 3.15 – Results when the approach, still trained on KITTI, is applied on other urban scenes datasets (from top to bottom: Driving ([111]), Cityscapes ([112]) and pictures taken in Rennes). From left to right: **a)** Input image. **b)** Network prediction. **c)** Estimated disparity map. **d)** Estimated confidence map (yellow means low-confidence).

We can indeed see that our approach, while trained on KITTI only, is able to return plausible disparity maps and to synthesize convincing new views from a synthetic dataset (Driving [111], first row), another automatic driving dataset (Cityscapes [112], row 2) and from images captured in natural conditions using a smartphone (Rennes, row 3).

The network was also trained on other datasets, with variable baselines and semantics (such as the small-baseline light field dataset Flowers [90] and the 3D movie database

Hollywood [113]), with convincing visual predictions. Visual results on these datasets are shown in figure 3.16 and in the supplementary video (`http://clim.inria.fr/research/MonocularSynthesis/monocular.html`).

Figure 3.16 – Images produced when the network is applied on other datasets (Hollywood [113] and Flowers [90]). From left to right: left image produced, input image, right image produced.

3.7.8 Pushing the semantics

In this section, we are interested in studying the limits of the usable semantics for our network. We know that our method cannot work in a fully generic setting, with any random semantics, due to its training procedure. We also know that our method can be used efficiently on urban scenes. We are interested in evaluating how generic it is on a variety of related, but different scenes. We choose one of them to illustrate the strengths and the limits of our approach, in figure 3.17.

We searched for toy car images online and applied our method on them. We want to see if our method trained on urban scenes is able to extend to toy cars, similar but different objects.

We notice that our method is able to extend efficiently to different elements (even though the result is not as efficient as it is for urban scenes). We also note that the geometrical configuration of the considered scene is an important factor in its quality; working from images with significantly different geometrical configurations from the training set usually leads to unsatisfying results for our method.

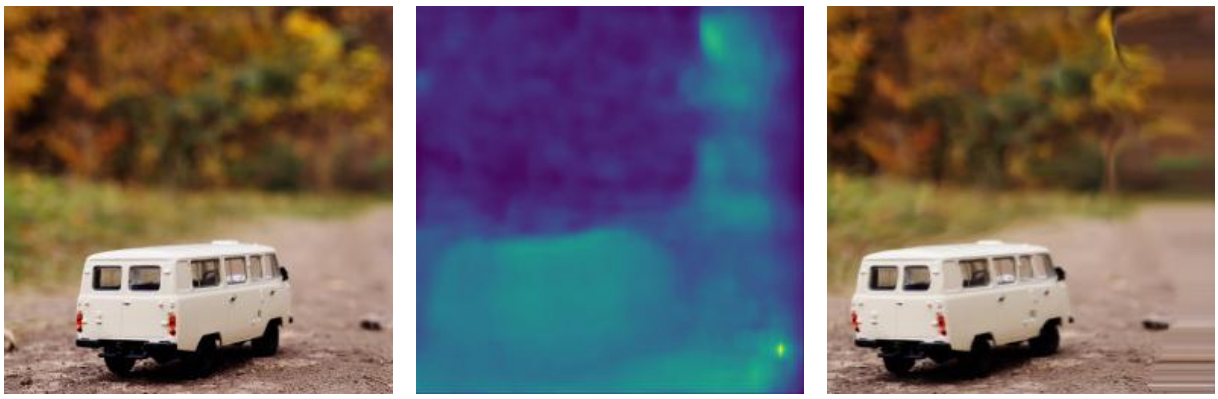

Figure 3.17 – Result when the method is applied on a toy car scene. From left to right: input image, estimated depth map and prediction

3.7.9 Limits of the approach

If the approach is able to perform convincing view synthesis in many cases, limits have to be highlighted. Since the geometrical refiners are all based on the color gradient, it means that they notably have a hard time segmenting unusual structures for which the color greatly varies (see figure 3.18). It is noteworthy, though, that since the network is not able to process this region correctly, it classifies it as a low-confidence region.

If our method shows an ability to be trained on a specific semantics dataset and tested on another dataset with similar semantics, it is far from being generic, and encounters difficulties when it has to predict images and objects that were never perceived before in the training set. For instance, in figure 3.20, we test our network trained only on urban scenes on a scene with totally different semantics; we conclude that the results are really distorted and unconvincing. It comes as no surprise and is totally expected, due to the requirement of pre-defined strong priors we had mentioned, and this property is true of all state-of-the-art monocular methods when trained on KITTI and tested on this image. This is still an important point to mention: the network can be extended to similar semantics and geometrical configurations, but faces a very significant struggle when the input element drastically varies from what it was used to perceiving during training.

As a whole, we can also note that the method also encounters difficulties when working on thin structures in front of a background with a strong color gradient (see figure 3.18).

Finally, in some situations, notably when faced with significantly textured contents,

the method can output a final image with good quality, but based on erroneous depth estimation. Indeed, as shown in figure 3.19, when working with pixels with a similar color, several possible depth maps, and even totally inaccurate ones, can lead to a similar image synthesized in the end. It should be noted that those depth estimation errors are mostly present when working on some natural images.

The problem of boundary pixels

When changing viewpoints and simulating a camera displacement, new areas, totally absent from the image, and which can hardly be guessed most of the time, have to be produced near the boundary. We notice that our Refiner is not able to come up with something efficient in these regions; it is mostly geared towards inpainting occluded regions within the image. Using our depth estimation process, we can easily spot the boundary pixels corresponding to entirely new regions that will have to be generated in the novel view. We then decide not to set these pixels as low-confidence, but instead only encourage the confidence map regularity relative to their neighborhood in these new regions. This constraint makes sense because we know that our Refiner not able to design a new region from scratch faithfully. In these regions, the problem is more tricky, and we have the idea that considering adversarial-based methods can be a way to improve the appearance in these areas. This is one of the main motivations going into the next chapter.

Figure 3.18 – Failure cases for our approach. From left to right: Input image, ground truth image, network prediction, estimated disparity map, confidence map.

Figure 3.19 – Another example of view synthesis result using our network on natural images captured using a smartphone. From left to right: input image, prediction, depth estimation. We can note that the depth estimation is inaccurate in the foreground, notably in surfaces of the same texture. This error in the depth estimation process has nevertheless few consequences on the finally synthesized image.

Figure 3.20 – Example when testing our approach, trained for urban scenes, on a semantics vastly different from the contents of the training set. We can see that, unsurprisingly, our approach is clearly unable to handle the problem.

3.7.10 Conclusion

We have presented in this chapter a supervised CNN-based approach able to perform monocular view synthesis. The MobileNet based-encoder allows us to obtain a good disparity-based prediction with a low number of parameters. This prediction is then refined in regions where artifacts are still present, occluded areas and mispredicted parts using the Refiner. The network is also able to estimate the confidence that it has in its own disparity-based prediction, and is able to identify the structures that it has not predicted correctly. The method outperforms state-of-the-art approaches metric-wise and visually in the domain of monocular view generation on the KITTI dataset. In the next chapter, we will focus on extending this work to light fields, and we will aim at improving the flaws of the method.

Chapter 4

Light field reconstruction from one single image

4.1 Introduction

In this chapter, we tackle a specific problem: to synthesize an entire light field from one single image. This problem has a variety of applications, such as generating several views of a scene, extracting depth and automatically identifying occluded regions from images captured with regular 2D cameras.

We have seen in the previous chapter that working from one single image is a very challenging problem, for at test time, the approach lacks information, e.g. on scene geometry. The method hence needs strong priors on scene geometry and semantics. Learning-based methods are therefore very good candidates for these tasks, since priors can be automatically learnt from data. In this chapter, we describe a method that is able to produce an entire light field, estimate scene depth and identify occluded regions from just one single image. This way, we can benefit from light field features without requiring a light field capture set-up, e.g., simulating perspective shift and post-capture digital re-focusing.

We propose a lightweight architecture based on what was described previously, but enhanced to be able to generate an entire light field and to better handle occlusions using an adversarial approach. The network is trained on pairs of images and learns to perform a forward and backward view synthesis, with two independent branches, thanks to the estimation of two disparity maps. Checking the consistency of the two independent predictions allows us to identify occluded regions and compute a disparity confidence map. At test time, the network only needs one image to compute the two disparity maps that are then used to identify the occluded regions. This disparity confidence map is used to control the application of an adversarial technique for occlusion handling. We show that

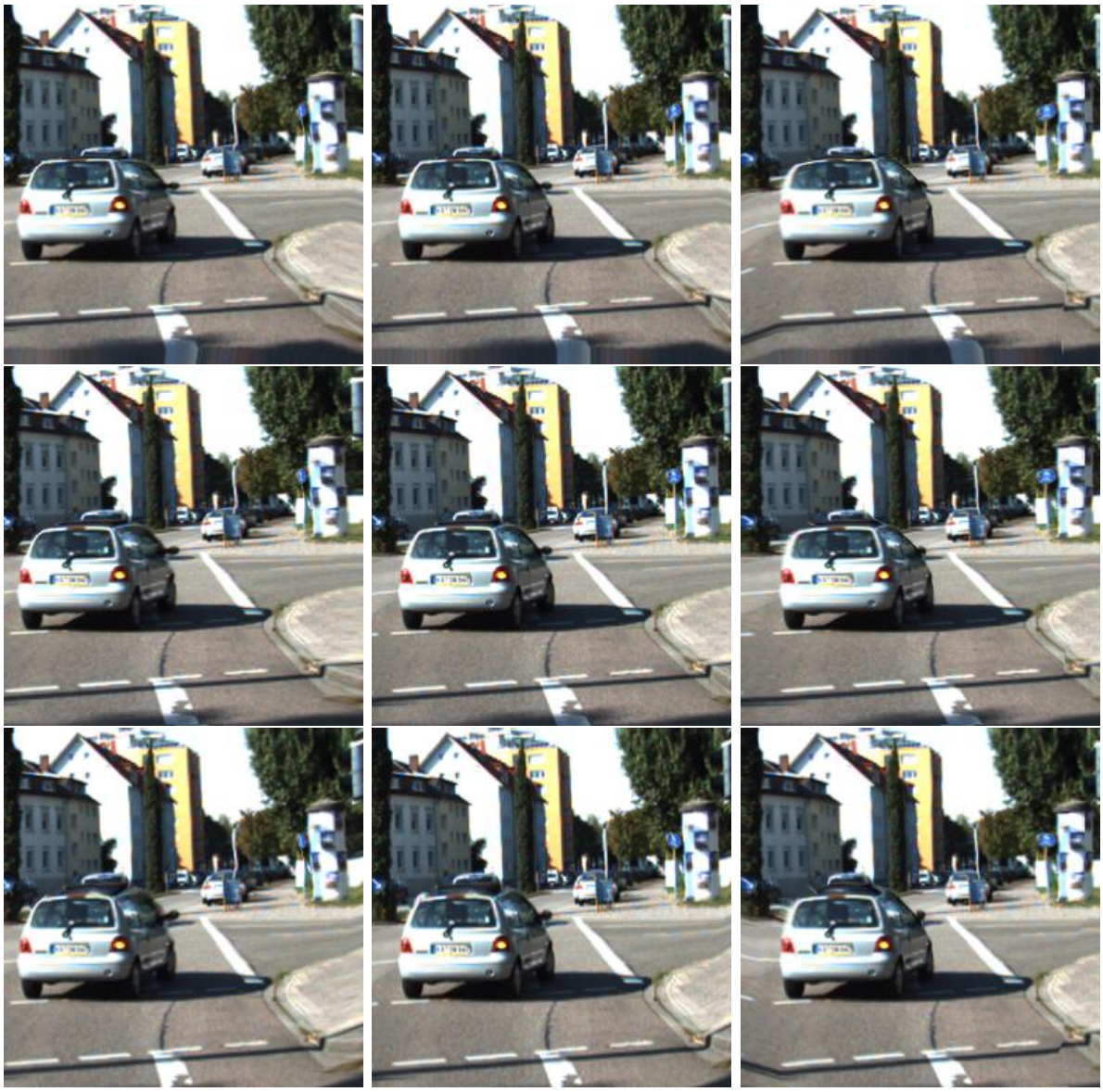

Figure 4.1 – A light field generated from one single image (input is the central view in the figure). The approach is trained on KITTI stereo contents, and is augmented using our method at test time to generate the light field.

the network can be trained on light field data, and that it outperforms reference techniques trained on light field datasets, such as [90], in terms of reconstructed light field quality.

Now, training on light field data as in [90] necessarily restricts the scope of the approach, as this requires a large amount of data that is not easy to capture. Besides, such monocular approaches are also bound a lot by semantics of the training data, making it hard to train a network that can be usable for a variety of scene geometry and semantics, unless a sufficient number of examples of diverse scenes is present in the training set. Existing light field datasets are in general too limited to meet that requirement. We show that the proposed architecture can be trained on stereo content. This drastically increases the amount of possible training data that can be exploited by our approach. We show that the proposed network produces plausible and good-quality light fields even when trained from stereo images with large baselines as in the KITTI dataset ([100]), and this way produces light fields with large fields of view.

In summary, our contributions are:

— A lightweight neural network based on the work described in the previous chapter, extended to be able to generate a full light field from one single view, with occlusion handling relying on both a computed disparity confidence map and an adversarial approach.

— A light field synthesis method from one single image that not only outperforms reference methods when trained and tested on light field data, but that can also generalize to much more diverse scenes, thanks to its ability to be trained on stereo datasets. The method hence enables convincing light field features (e.g., digital refocusing, virtual camera motion) from only single 2D images.

4.2 Context and objectives

4.2.1 Motivation

The main objective of this second contribution was to extend the work from the previous chapter, carried out on stereo data, to light field data. This may seem like a straightforward extension, given that light fields can be efficiently represented using the

central view and a depth map. Hence, once the depth map is computed and the process of view prediction is carried out, the problem seems to be similar. Besides, if we have access to light field data, just using the Disparity-Based Predictor to estimate a 2-channel disparity map seems like an obvious and efficient extension to light fields. There are nonetheless significant obstacles that make this extension not as straightforward:

— First, as specified in the previous chapter, the method relies on the existence of a dataset with enough similarity in both semantics and geometrical configuration. If these datasets are relatively easy to find for stereo data, they are much harder to capture for light fields. In this framework, the problem of generality is then really more significant, because it is much harder to find relevant datasets.
— The method described in the previous chapter performs an interpolation process to generate intermediary views between the input and the target images. This interpolation process is not without flaw; and it is crucial to make it more precise when working on light fields, given a 2D range of consistent images needs to be estimated at this time.
— The confidence process also needs to be updated with the change in nature for the input images available during training.

Another objective of this new contribution is to improve over the flaws of the method described in the previous chapter. Notably:

— We are interested in improving the quality of reconstruction for occluded regions, and in particular for areas near the boundary of the images. We want to investigate whether adding an adversarial component to it can prove to beneficial.
— Our confidence map estimation has several flaws. First, it is not 100% accurate, for it usually includes all poorly-reconstructed or non-Lambertian regions together; we believe that there is a way to be more precise in the definition of regions to be processed. Besides, the confidence training method is efficient, but makes the training process a bit complex, and we are also eager to simplify the whole process. Doing away with this process is also a way to reduce the number of parameters of our overall method. Finally, our confidence map is, from an implementation viewpoint, directly linked to the network, and has a graph connection with the result produced by the Refiner, meaning that our network can not use ground truth, or handcrafted confidence maps to generate its final result. We aim at tackling these problems with our new process.

4.2.2 Addressing the lack of generality in semantics

Monocular view synthesis usually requires a dataset with strong similarities in geometrical configuration, as well as in semantics. Such datasets are commonplace and easy to capture for stereo contents, but are more tricky to capture as light fields. The reference light field dataset for monocular view synthesis is Flowers ([90]), a dataset with rather simple semantics, few depth planes within scenes and with a rather small baseline. Evaluating our method on this dataset only does not seem sufficient to validate it. We were thus interested in evaluating it in more diverse and challenging datasets.

When training and evaluating our network on more generic datasets such as the one first presented in [29], we noticed that our method struggled. In particular, we noted a strong correlation between the presence of one class of objects in the training set and the efficiency of the method on the same kind of object on the test set. In other words, significant gaps of performance could be noted between two views of the same test set, notably depending on the semantics of the image. We felt like it was then not totally satisfying; in the previous chapter, our methods also had limitations as for which kind of semantics they could be applied on, but those limitations were well-known and easy to delimit. Here, the strong dependency in the contents of the training set made the analysis far more complicated. Adding more elements to the training set is then of course a potential valid answer: theoretically, the more elements the method has processed during training, the better it will be for more generic use. Capturing a totally generic database of light fields is nevertheless a difficult process, and for that reason we decided to follow another route, and instead tried to harvest results that were achieved for view synthesis in non-light field frameworks.

To address this concern, we notably focused on considering a proxy task. The key idea was to begin the analysis with a "guidance" network that has managed to address a proxy task in the most generic way possible. To do so, we studied the field of image segmentation, and tried to combine a segmentation-based method with the guidance network strategy through self-supervised learning, as described in [114]. Proceeding this way would have allowed us, hopefully, to add a guidance network already trained for these tasks on our own network, to tackle our own problem. Experimentally, we have found that it was very difficult to find methods and proxy tasks that were able to obtain good performance on a wide variety of semantics, when it is trained from one single image. We thus decided to

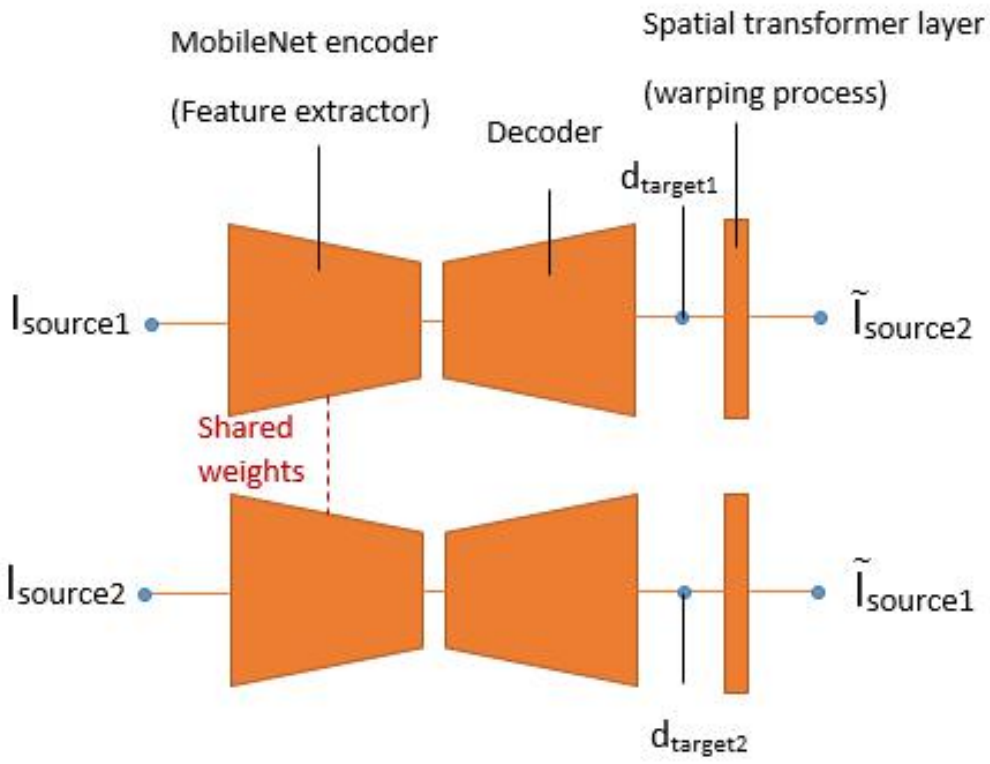

Figure 4.2 – Outline of the DBP section of our architecture

follow a slightly different route: using stereo contents as a proxy task, to generate a full light field. Given stereo datasets are easier to capture, we believe this is a good way to go beyond the semantics of light field data, all while harvesting the positive contributions presented so far.

4.3 Description of the method

While the proposed method builds upon the architecture presented in the previous chapter designed for generating new views from one single image in a stereo setting, it is extended here in order to be able to generate an entire light field from one single input view. In addition, the network is designed in such a way that it can be trained either with stereo content or using pairs of light field views. While the network can be trained from stereo content as well as from pairs of light field views, when using classical stereo content to train the network, the pipeline is adapted to account for naturally missing information,

e.g., related to scene geometry, through the resort to the Refiner, as explained in sections 2.3 and 2.5.

4.3.1 Disparity-Based Predictor (DBP)

The **Disparity-Based Predictor (DBP)** is a neural network made up of two branches, accounting for both the Feature Extractor and the Decoder, as shown in figure 4.2. It receives one single input image and estimates a disparity map. Trained with a pair of views, the two branches of the DBP take one image of the pair as input, and consider the other image as ground truth. In each branch, the Feature Extractor is used to extract features of the input image, using a MobileNet architecture, with weights shared with the other branch. The weights are initialized with ImageNet ([105]) weights. A second part in each branch, the Decoder, produces a disparity map through upsampling layers and using skip-connections. Finally, a spatial transformer layer is used to warp the disparity map to predict a view. This first prediction is based on the warping of pixels, hence the result is usually sharp, but artifacts may remain due to disparity errors, in particular in occluded regions.

4.3.2 Estimating the prediction confidence

The next step consists in identifying the regions not well handled by DBP and the warping process. In order to compute the confidence we have in our first prediction, we use the already trained DBP, and we follow the protocol defined in figure 4.3. We send as input of our two branches the same input image I_{source}. This will give us two independent predictions in disparity, centered on two different target views ($d_{target1}$ and $d_{target2}$). We then re-warp these disparities back onto the source view (giving us $d_{source1}$ and $d_{source2}$), and we take as confidence measure C_γ their difference, using the following expression:

$$C_\gamma = \exp(-\gamma |d_{source1} - d_{source2}|) \tag{4.1}$$

We can note that in contrast with the method presented in the previous chapter, the error is directly computed and not estimated using a trained network. Doing this simplifies the learning process, and allows us to reduce the number of parameters. It is also a way to improve the confidence map, so that occluded regions are better identified, as shown in the Results section.

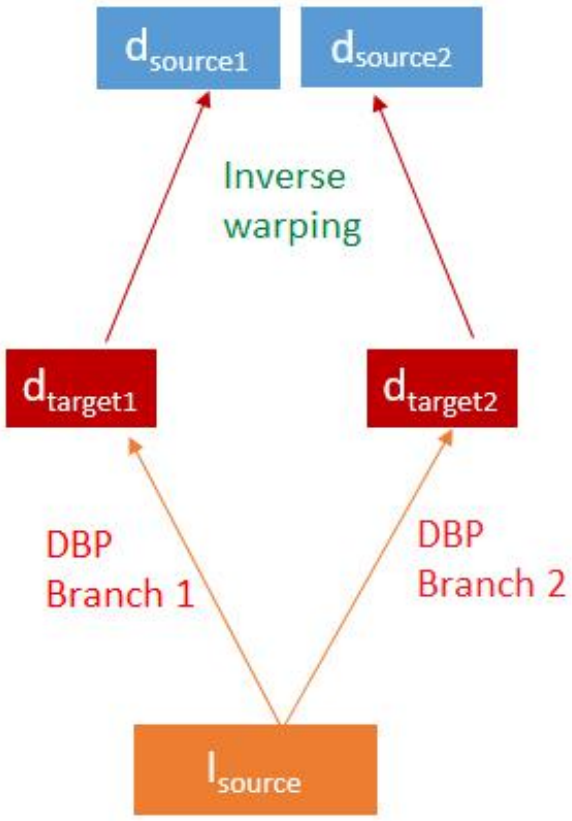

Figure 4.3 – Diagram depicting the employed confidence method.

4.3.3 Refiner based on a GAN

To correct errors in lower confidence regions, and to account for the fact that the corresponding information is not available at test time, we use a **Refiner network** trained using an adversarial loss combined with a pixel-wise metrics. This leads to plausible estimates of the pixels in the occluded regions. The refiner network is actually the generator of a Wasserstein GAN ([52]), and adversarial learning is carried out only in regions of low confidence.

The refiner is built as an encoder-decoder structure with skip-connections. It is made up of a succession of Spectrally Normalized convolutional layers (as first described in [53]). The discriminator is also built using these layers. To make sure that the learned distribution remains faithful to the input and ground truth data, we also add pixelwise and gradient-wise metrics besides the Wasserstein loss. This allows us to fill occluded regions with synthesized contents, which will be both realistic (thanks to the adversarial loss) and as faithful as possible (thanks to the pixelwise metrics).

At test time, we only use the generator part of the adversarial process to synthesize our view. It takes as input the warped prediction, as well as the estimated disparity map. The final predicted view V_{fin} is obtained by combining the two predictions using the confidence map as

$$V_{fin} = C_\gamma V_{disp} + (1 - C_\gamma) V_{ref} \tag{4.2}$$

where V_{disp} is the output of DBP and an input to the refiner, and V_{ref} the output of the refiner, and C_γ the computed confidence map.

The method can be tailored to be efficiently trained on both light fields and stereo content. There is one refiner per branch, which is applied in both learning and test time.

4.3.4 Training on light fields

When using light fields for training, we have access to both horizontal and vertical disparities, hence the DBP can be trained to estimate these two disparities and produce the corresponding horizontal and vertical warpings.

We extract pairs of views by taking the center view as one of the two images of the

pair, and the other one randomly within the light field. The maximum disparities of the light field are taken as reference.

When working on views which are not extreme, and assuming a regular sampling of views in structured light fields, we estimate the disparity d_{int} of an intermediate view by interpolation as

$$d_{int}(x) = \alpha d(x - (1 - \alpha)d(x)) \tag{4.3}$$

where α represents the targeted position, and x the bidimensional coordinates. This allows us to obtain an interpolated disparity map for warping, that will tend to favor background disparity for occluded regions, and lead to more plausible results than when simply multiplying the disparity map.

4.3.5 Training on stereo content

The method can also be trained on stereo data, and be used to generate full light fields. In this case, we can only train the method with horizontal disparity, and have to infer vertical disparity at test time. We therefore add a simple module to infer the vertical disparity at test time, once the network was trained on stereo contents. The new, two-channel disparity map d_{new} is obtained from the horizontal, predicted one, by applying the following transformation to the horizontal disparity map d_{hor}:

$$d_{new}(y, x) = \alpha d_{hor}(\alpha_y d_{hor}(x), x - (1 - \alpha_x)d_{hor}(x))) \tag{4.4}$$

where y accounts for vertical coordinates, while x accounts for horizontal coordinates, and $\alpha = (\alpha_y, \alpha_x)$ is a set of parameters accounting for the relative position of the requested view relatively to the input view. Given that the warped disparity map may however contain errors especially in the foreground near the borders of the image, we improve it by applying an auto-regressive extrapolation along the vertical lines and from the 50 previous points. The rest of the network proceeds with the warped prediction, and refines and automatically improves the occluded regions at test time.

4.3.6 Summary

In summary, the procedure is as follows:

— From a pair of images, learning the disparity and warping from it through the DBP to generate one from the other.
— Through a confidence computation obtained by inputting the same image in both branches, determining which regions are likely to be accurate.
— In the regions with low-confidence, using a refiner with adversarial learning to improve the results.

4.4 Learning procedure

Let L_{DBP} and R_{DBP} be the DBP-based predictions, and L and R the ground truth images, and d_L and d_R the disparity maps for the warping towards predictions L and R. We first train the DBP using the metrics:

$$\lambda_0(||L_{DBP} - L||_1 + ||R_{DBP} - R||_1) + \lambda_1(||\nabla L_{DBP} - \nabla L||_1 + ||\nabla R_{DBP} - \nabla R||_1) \quad (4.5)$$

Before training the Refiner, we add a step of geometrical restructuring for the DBP.

Finally, we freeze the weights of DBP, and train the Refiner in order to minimize the loss function

$$\lambda_4(||L_{REF} - L||_1) + \lambda_5(||\nabla L_{REF} - \nabla L||_1) + \lambda_6(||L^* - L||_1) + \lambda_7(||\nabla L^* - \nabla L||_1 + \lambda_8 \mathcal{L}(L^*, L) \quad (4.6)$$

where L_{REF} is the prediction performed by the Refiner, L the ground truth image, L^* the final combined prediction $L^* = C_\gamma L_{DBP} + (1 - C_\gamma) L_{REF}$, and $\mathcal{L}$ the Wasserstein loss. The discriminator for the adversarial process is trained using only this Wasserstein loss. For the hyperparameters, we consider: $\gamma = 0.08$, $\lambda_0 = 0.80$, $\lambda_1 = 0.20$, $\lambda_4 = 0.27$, $\lambda_5 = 0.054$, $\lambda_6 = 0.54$, $\lambda_7 = 0.135$, $\lambda_8 = 0.01$. We optimize our approach using the Adam algorithm ([47]), with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a learning rate of 0.0001 for the overall network (with 0.00001 for the discriminator during training). The work was implemented using TensorFlow ([108]) and Keras ([109]). The network was stopped when no improvement in the validation metrics was obtained after 20 epochs. The network is fully trained after only a few hours, and contains around 6 million parameters at training time.

For the following experiments, our method for training took as input patches of resolution 256×256 (for the stereo case) or 256×512 (for the light field case), normalized

between -1 and 1, with data augmentation in 20 % of the cases, with random gamma and brightness transformations. In this chapter, we used two datasets for comparison: **Flowers** ([90]) and **KITTI** ([100]). **Flowers** is a light field dataset, with rather small baselines, comprising around 3,000 light fields of flowers in similar geometrical configurations. We systematically pick the central view as one element of the pair, and we randomly choose another view as the other element of the pair. We adjust the value of α to account for the coordinate of the selected view. As a starting point, we only focus on one corner view as target that we arbitrarily choose as the reference disparity ($\alpha = (1, 1)$). After 10 epochs, we add the rest of the views as possible target views and the interpolation process described in section 4.3.4 is then applied. We perform a random train-test-validation split, to be able to compare our approach, with 100 elements from the dataset randomly chosen as test set, as in [90]. **KITTI** is a stereo dataset which depicts urban scenes, and contain pairs of images with a very significant disparity gap between them. In this work, we use the same split as in the previous chapter.

4.5 Evaluation

We compare the proposed approach to several methods: LF4D ([90]), a method able to predict a full light field from one single image, by enforcing epipolar constraints within the predicted light field, using the code provided by the authors. We also compare visually our approach with the method in [87], in the stereo case, using the network provided by the authors. We also compare our method to the method from our previous chapter, and with the reference method [89], both focused on working in a stereo setting. To evaluate our stereo-based approach, we also use it on Flowers by only training it from 2 aligned views on the central line of the light field ([90]). For evaluation, we use PSNR, SSIM and LPIPS ([104]) as reference metrics. Due to the visual nature of the task, we strongly recommend the reader to take a look at the Supplementary video (`http://clim.inria.fr/research/MonocularSynthesis/supplementary.html`), which displays other examples of views synthesized using the proposed method.

4.5.1 Light Field View Synthesis Results

Training and testing with light field data We first focus on training and testing the network with light fields. For that, we use the Flowers dataset ([90]). We evaluate predicted

PSNR/SSIM	Ours	LF4D	Stereo
4 corners	**34.97/0.94**	31.61/0.89	33.54/0.93
Full LF	**38.41/0.96**	35.10/0.94	37.16/0.95

Table 4.1 – Statistical comparisons between our method trained on light field data (Ours), reference method LF4D ([90]), and our stereo-based method (Stereo). We display the mean PSNRs and SSIM on the 4 corner views (the most difficult ones to predict), as well as on the full light field.

Figure 4.4 – Visual prediction for a top-left image from the Flowers test set, as well as the corresponding L1 errors, for, from left to right, our method, LF4D ([90]) and the stereo version of our method. The errors were multiplied with a factor of 3 for better visualization.

views in comparison with the reference method LF4D ([90]), in figures 4.4, 4.5 and 4.6, in table 4.1, as well as in the supplementary video (`http://clim.inria.fr/research/MonocularSynthesis/supplementary.html`). We see that our approach clearly outperforms LF4D, both metric-wise and visually.

We also use the Flowers dataset to evaluate our stereo-training based approach, i.e. by only training the network on stereo aligned pairs (extreme left-side view - center view and center view - extreme right-side view). The results (the last row of figure 4.4 and table 4.1) show that our method, even when trained on stereo content, manages to outperform the LF4D monocular light field synthesis method, and is able to produce high-quality light fields. This shows that our stereo to light fields adaptation module is very efficient.

Training on stereo content We also train the network using the stereo KITTI dataset ([100]), in order to build a full light field. The views produced have no ground truth equivalent; only visual evaluation is possible in this case. Visual results are shown in figure 4.1

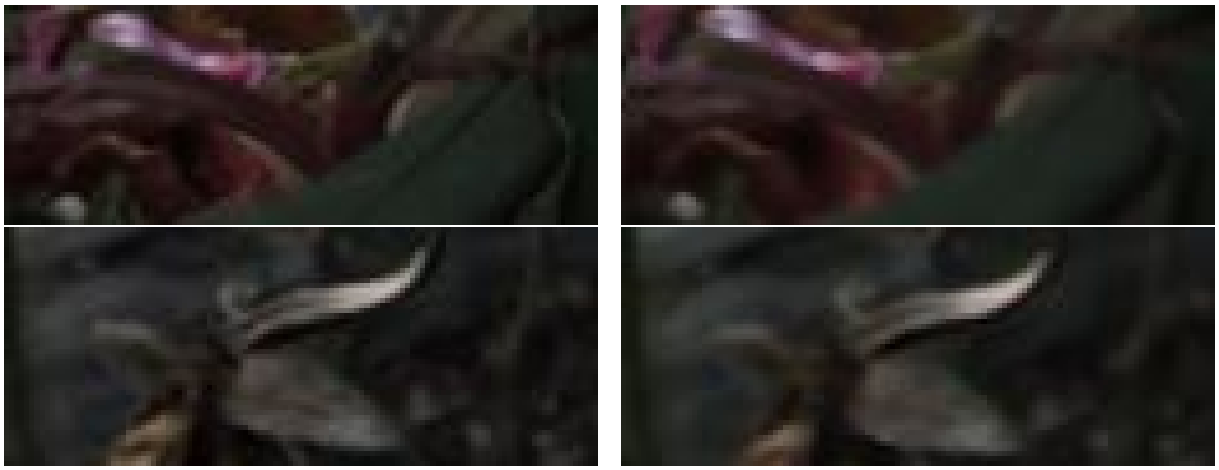

Figure 4.5 – Close-up views from figure 4.4. On the left side, our results, on the right side, the results obtained in [90]. We note that our results are sharper and structurally more consistent.

Figure 4.6 – Supplementary visual comparisons between our work (left-side) and [90] (right-side). We note that our images are sharper and better quality.

Figure 4.7 – Visual comparison of two of our predictions with Sun's method, for similar geometrical transformations (from left to right, 2 sequences of: input, our prediction, and the prediction obtained from [87]). The views we produce are less blurry and have fewer distortions.

and in the supplementary video (`http://clim.inria.fr/research/MonocularSynthesis/supplementary.html`). To evaluate our approach, we compare it visually to the monocular part of the method in [87]. The network, also trained on KITTI, receives as input one image and a transformation vector expressing the relative coordinates of the target view. We specify to the pre-trained network a transformation vector similar to ours.

We note that our approach clearly performs better visually on this data (see figure 4.7). This is probably because the vertical transformations are not present in the KITTI training set, and can thus not be learnt efficiently by the method in [87]. Given that our approach is optimized to generate the light field, we are able for this task to obtain more realistic results.

To evaluate metric-wise our predictions, we also compare them with stereo-based view

KITTI Test Set	PSNR	SSIM	LPIPS
Ours	**19.96**	**0.76**	**0.130**
[115]	19.24	0.74	0.139
Deep3D ([89])	19.08	0.74	0.220

Table 4.2 – Comparison of the results of our approach with 2 reference methods ([115], [89]) in a stereo setting. For PSNR and SSIM, the higher, the better. For LPIPS, the lower, the better.

synthesis methods (including ours) and [89] in table 4.2, on the KITTI test set, in a stereo setting. We note that our approach significantly outperforms these two methods for the 3 chosen metrics. We can note that we obtain those results with a smaller number of parameters (notably, 200,000 fewer parameters than what we had in the previous contribution). We show in figure 4.8 a visual stereo prediction, associated with the L1 error. We can see that the predicted view is rather high-quality.

Finally, we compare our confidence computation process with the one described in the previous chapter in figure 4.9. We note that our occlusion identification process is significantly more efficient.

Testing on natural images We can also test our network on natural images, captured using a smartphone. It allows us to produce a full light field from one single image. A visual example of it is shown in figure 4.10.

4.5.2 Ablation study

Impact of the confidence-based refiner We evaluate the impact of the refiner on the result in tables 4.3 and 4.5. We can note that it increases the performance both in PSNR and SSIM for both datasets. Its contribution is, though, more significant when working on KITTI, due to its more significant occluded regions. We also evaluate its positive contribution when training the approach on stereo contents, and using it to generate light fields in table 4.4. We note that the Refiner in this case also allows to significantly improve the performance of the approach.

Impact of adversarial learning We also evaluate the impact of our adversarial process on the result. We note that depending on the chosen dataset, we do not draw the same conclusions. When working on Flowers (see table 4.3), we note that the adversarial

Figure 4.8 – Result of our approach in a stereo setting, on the KITTI test set, for evaluation. From top to bottom: input image, our prediction, ground truth image, L1 error.

Flowers	**Ours**	**No AL**	**No Refiner**
PSNR	**38.41**	38.40	37.59
SSIM	**0.96**	0.96	0.95

Table 4.3 – Statistical comparisons for the ablation study on the Flowers test set. No Refiner only uses the warped prediction, No AL does not use adversarial learning

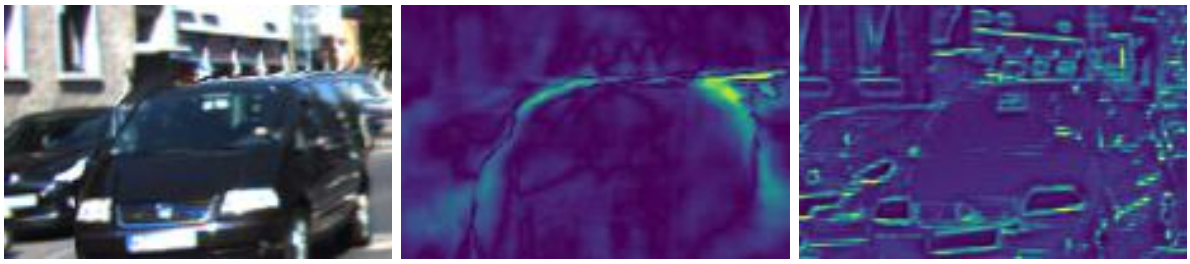

Figure 4.9 – Visual evaluation and comparison of the confidence map. Yellow means low-confidence. From left to right: our prediction, confidence map returned by our approach, confidence map returned by [115] in the same setting. We note that our way to compute the confidence map is significantly better at specifically capturing occluded regions.

Flowers	**Stereo ours**	**Stereo No refiner**
PSNR	**37.16**	36.02
SSIM	**0.95**	0.93

Table 4.4 – Statistical comparisons for the ablation study on the Flowers test set. Stereo ours is our stereo-based light field synthesis method, Stereo No Refiner evaluates the prediction when no refiner is used.

KITTI Test Set	**Ours**	**No AL**	**No refiner**
PSNR	**19.96**	19.85	18.87
SSIM	**0.76**	0.75	0.72
LPIPS	**0.130**	0.135	0.144

Table 4.5 – Statistical comparisons for the ablation study on the KITTI test set. No Refiner only uses the warped prediction, No AL does not use adversarial learning.

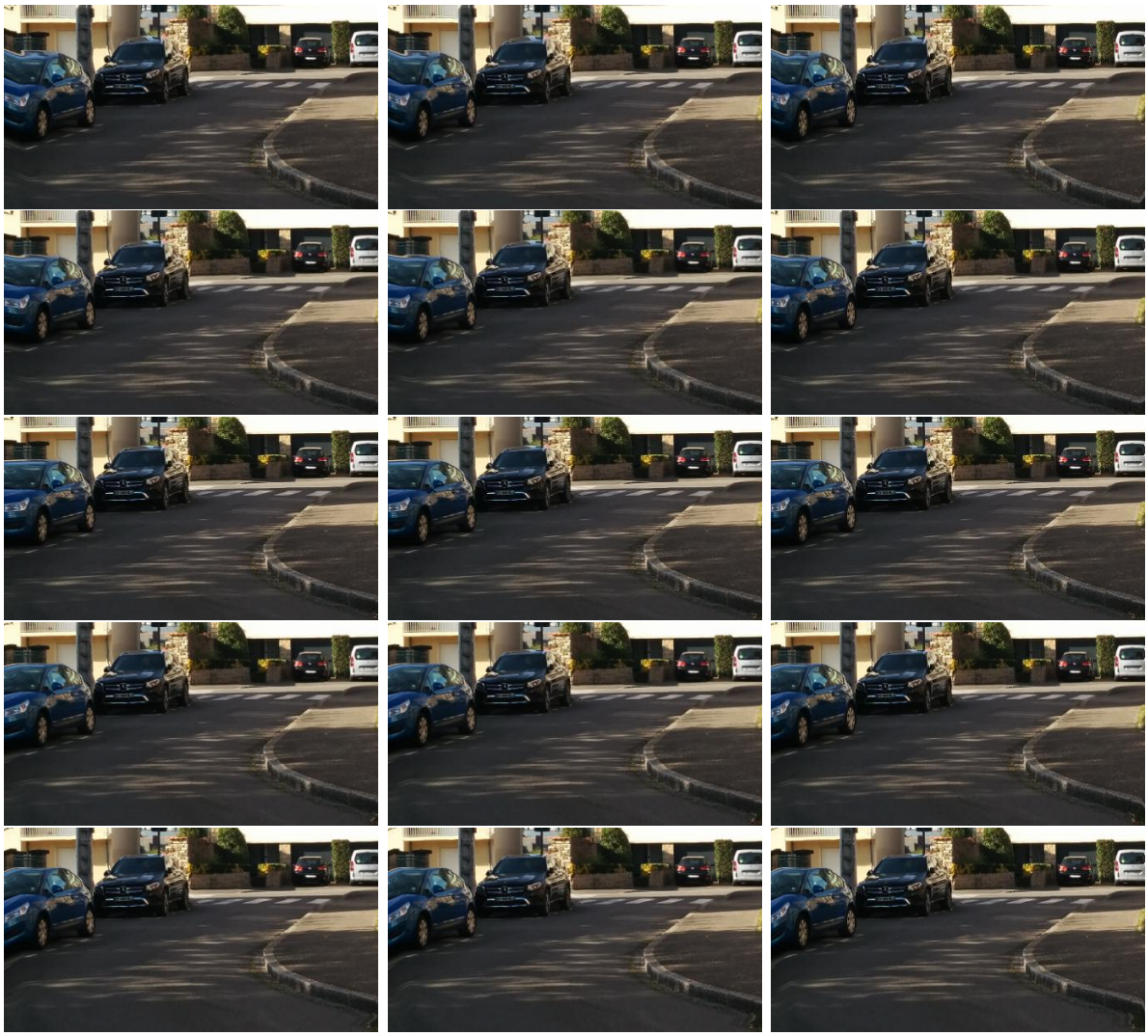

Figure 4.10 – A light field generated from one single image (input is the central view in the figure). The approach is tested on a natural image, captured using a smartphone. For a result with higher resolution, we advise the reader to check the supplementary video at `http://clim.inria.fr/research/MonocularSynthesis/supplementary.html`.

process does not really have a significant impact. The occluded regions in Flowers are indeed smaller and then easier to fill, reducing the usefulness of the adversarial loss.

On the other hand, when working on KITTI, we can see that the adversarial process is much more beneficial, giving an overall increase in PSNR and SSIM, but more importantly a significantly better LPIPS ([104]), showing that it is an adequate way to improve the perceptiveness of our images.

4.6 Further analyses and experiments

4.6.1 Analyzing the interpolation process

Between the previous chapter and the current one, one emphasis is put onto the interpolation process: instead of just multiplying the disparity with a value, we instead warp the depth map onto the desired position before using it for warping. The interest of such an evolution is to make sure that occluded pixels are treated as background instead of foreground when considering interpolation. The positive contribution of this updated interpolation process is shown in figure 4.11.

4.6.2 Analyzing the confidence measure

Compared with the previous chapter, we have updated here our confidence measure. Our new confidence measure shows several benefits when compared with previously:

— It is not directly part of the training process: hence, the method requires fewer parameters, and we can also make sure of its accuracy, while the self-learnt confidence measure can at times be flawed.

— We noticed experimentally then when training the method from the previous chapter, the graph connection between the confidence map estimator and the refiner was central to the overall performance; removing it led to significant decrease in performance. For that reason, we feel like it is relevant to try and do away with this limitation, hence the point of this new confidence measure, which is not part of the graph itself.

— We change the overall idea behind the confidence map computation; indeed, in the previous chapter, the error was computed on disparity inconsistency positions. The underlying idea behind such a computation was to identify regions where forward-

Figure 4.11 – Showing the improvements of the new interpolation method. Top line: interpolation method from chapter 1, bottom row: interpolation method for our current method. We note that near occluded regions (near the gray car), the interpolation process is much more accurate for the bottom line. From left to right, interpolated views with increasing distance from the source view.

backward errors in position were strong, given they are more likely to be regions with a significant depth change that has not been addressed. In this work, we have decided to adopt a error measure based instead on disparity magnitude; instead of comparing the relative positions when doing forward-backward, we compare the actual disparity values on the same pixel. This is an efficient process to accurately identify occluded regions.

— Finally, one of the reasons why we adopted this new confidence measure is because it practically worked: by having a more accurate error estimation, we can use the Refiner in more relevant regions, allowing an improvement both in PSNR and LPIPS.

Another change we brought was the architecture of the Refiner (in order to produce efficient adversarial training), which moved from a simple neural network with successive convolutional layers to a relatively shallow encoder-decoder structure, with roughly the same number of parameters. We also noticed that this change in architecture improved our results, as shown in the ablation study.

We have suggested in this chapter a confidence measure relying on inconsistent depth estimation. We can also note that it is entirely possible to use a different confidence measure if we want the Refiner to be applied in specific regions. Our own experiments have shown that we can even handcraft our own confidence measures on the original prediction, which allows us to perform a relatively free and efficient inpainting process on the desired region.

4.6.3 Analyzing the adversarial process

We've found experimentally that the adversarial process helped significantly in handling occluded regions, and in particular in regions near image boundaries; indeed, these regions are now systematically classified as low-confidence, and require a full synthesis from neighbouring elements. The GAN helps us tackle this problem more efficiently as shown in table 4.5.

This adversarial process is nevertheless faced with limits, mostly related to the training process. Indeed, in spite of many efforts to favor training stability (through the resort to Wasserstein GANs, normally spectralized layers, giving more training iterations for the Discriminator than for the Generator...), we ended up very often in situations where our adversarial training could not converge.

Besides, we have also found experimentally that a very cautious and careful setting of hyperparameters has to be carried out on the validation set. Indeed, our adversarial process is prone to two extreme behaviors when the choice of hyperparameters is not valid. We have seen that putting a too high emphasis on the adversarial process usually led to strong and clear mode collapse; an extreme case of it is shown in figure 4.12 for a constant distribution. In this situation, the Generator managed to trick the Discriminator with a trivial distribution. On the other hand, setting a too small hyperparameter value for the adversarial process usually led to a negligible influence for it, given that the adversarial hyperparameter then became irrelevant metric-wise when compared with the pixel-wise metrics. Finding the right hyperparameters on the validation set, as well as having a right convergence usually led to a result that could be perceived as a thin line between mode collapse and absence of influence for the adversarial component.

This training difficulty has, in our opinion, strong consequences regarding its wide and

Figure 4.12 – Illustration of a mode collapse for our Adversarial Process, notably due to a poor choice of hyperparameters. We can note that low-confidence regions are only darkened.

generic use for many datasets. Indeed, we have found experimentally, that keeping the same set of GAN hyperparameters for other datasets did not lead to conclusive results; they had to be totally re-set. It is noteworthy that for all the other elements we have shown so far, the results were obtained while keeping the same hyperparameters for all datasets. We have then found that resorting to GANs, indeed led to better results on KITTI, but it also reduced the generality and comfort of use of our method on new datasets.

That dependency regarding the dataset is not really surprising: the core idea behind the use of GANs was to learn the semantic distribution of the training set, hence why a change in training set and semantics can have a drastic impact on the efficiency of our Generative Adversarial Networks.

4.7 Conclusion

In this chapter, we have described a method able to produce light fields, with a training from both light field datasets and stereo datasets. The proposed method allows us to generate high-quality light fields, from only one single input image and for diverse images and semantics. We manage to achieve good performance for producing these light fields, and are able to use stereo data to produce light fields with a wider variety of contents and semantics.

Chapter 5

Recurrent Neural Networks for Light Field View Synthesis

5.1 Introduction

Light field imaging has been getting increasingly popular in recent years, due to the amount of information that a single capture brings about the scene such as in particular parallax and scene geometry. Indeed, unlike traditional 2D images, light fields sample light rays emitted by the scene, and can be seen as a 2D array of views captured from different angles. This enables a variety of applications, such as post-capture refocusing and depth estimation.

Light field content, though, is computationally expensive to process, hence the need for light field compression. Performing efficient view synthesis is a way to reduce the amount of required data. The aim of the approach described in this chapter is to generate a full light field from only a subset of views. However, most approaches for view synthesis are either bound to a specific disparity range, or to a specific number of input views.

In summary, our contributions in this chapter are:

— An approach resorting to Recurrent Neural Networks to perform light field view synthesis from a subset of views. The method is able to compete with reference methods in the field.
— A method that can efficiently produce new light fields from a variable number of input images.
— A network resorting to a *Multi-Plane Image* representation of the scene for view synthesis, and that can be adjusted to any chosen configuration and distribution of depth planes at test time.

5.2 Relevance of the problem

The main idea behind the contribution is to relax the constraints that were set in the first two contributions, by considering this time several input views during test time. We are nevertheless interested in keeping other constraints: we want the method to be able to work from scratch without requiring complex annotations in the training set, and we want to have as few parameters as possible.

Tackling this problem, our main objective is to consider a situation that can still be commonplace for real-world applications. Though, we add more constraints from the user side, because the whole process requires more than one single image at test time. Doing so, we also guarantee that we will be able to obtain better performance for view synthesis, and better genericity: indeed, when working from more than one image, the network usually has more information at test time to produce more satisfying results, given that in this case, it is entirely possible to build a solution that is not purely based on semantics. We mostly expect our network to find connections between existing views that it receives as input.

We also note that having several cameras or sensors is getting increasingly commonplace, notably in smartphones. Hence, at least 2 views of the same scene can easily be captured in real-world use cases. We thus believe that designing a method that can perform light field synthesis given a small subset of input views is also an interesting way to reach performance and genericity, while not adding very significant constraints for the user.

Furthermore, we analyse that most networks are learnt for a specific configuration and number of input views. This allows to optimize the results for a specific configuration and for a chosen setting. Here, we are interested in developing a method that can work given various view configurations as input, without requiring retraining. We believe that this problem is an interesting one, for it means that depending on the input signal that we send, the network will opt for different PSV transformations and thus adjust to different view configurations. In particular, the method can be used for a number of input views that can fluctuate between 2 and 4, using the same neural network. We believe that this is an interesting development.

Following recent work in the literature ([116], [63]), we develop an architecture building a MPI representation. In most of these works, the MPI structure of the scene has to be set as a prior during training, and needs to remain the same at test time. Besides, a constant distance is assumed between the considered depth planes representing the scene, which of course does not represent accurately all scenes; depending on the contents and the geometry of the scene, it will be favourable to add a higher concentration of depth planes in some key regions, and fewer depth planes in some emptier regions. Through our work, we are able to choose during test time the position of the depth planes in the MPI model without retraining. Indeed, given that we connect one specific loop of our architecture with one specific depth plane of the MPI representation, by changing the number of loops, as well as their associated depth planes, we can adjust at will the MPI representation at test time.

5.3 Preliminary research remarks

We have found experimentally that the problems of *monocular view synthesis* and *view synthesis* with several input images are vastly different. Indeed, many conclusions we were able to draw from working in a monocular case did not necessarily stay true when working from several input images. In particular, many architectures and concepts that we found were clearly failing while tackling a monocular view synthesis task ended up being relatively efficient when working from several input images. This fact was evidenced in **chapter 3** by considering the results using the architecture from [60], shown to be efficient for video frame interpolation, in a monocular use case.

This is not particularly surprising, and can be explained by the fact that the two tasks are actually very different from a learning viewpoint. In the case of monocular view synthesis, the network aims at extracting as much information as possible from one image, using already learnt data priors; the role of the network is then to perform scene analysis from one image as efficiently as possible. Given that the problem is ill-posed in this case, most of the effort for the network is focused on using and learning appropriate priors for the problem. On the other hand, when several images are available, the role of the network is to find associations between pixels belonging to different views. Hence, the necessity to learn priors from it appears not as significant; what matters in this case is for the network to be able to draw associations between the various pixels.

The two problems are then very different, and have different constraints and requirements.

Due to this difference in nature, we draw different conclusions, and believe that other architectures could be useful for the task. In particular, we draw inspiration from [117], and decide to resort to a LSTM architecture, due to its lightweightedness and its efficiency.

5.4 Description of the method

5.4.1 Description of the approach

We aim at generating a full light field from a subset of input views (between 2 and 4 views). In particular, we decide to model our scenes as *Multi-Plane Images*. The MPI representation of a scene is a good way to model it, by describing the entire scene with a collection of (α, C) couples, associated with every depth plane of the given scene. For every depth plane, α is the *visibility map* associated with this depth plane; it is a map of values in a [0,1] range, describing how visible the corresponding pixel at this depth level actually is. C stands for the color map, describing the color value of the pixel at this depth level. By accumulating and combining all the visibility maps as well as their associated colormaps, we can reconstruct the target image.

This MPI representation is usually arbitrarily chosen, and kept at test time. Most of the time ([64], [65]), the choice is made to have a constant distribution in terms of depth plane presence; we assume a given number of depth planes, equally distributed. This approximation is consistent, but is not perfect: indeed, some regions in the scene might benefit from having a more concentrated distribution of depth planes to model them, while some studied depth planes might not be relevant for the given scene. The constant distribution in terms of depth plane is thus suboptimal, and we would like to find ways to leverage other distributions.

Of course, the problem of choosing which distribution of depth planes should be chosen is a tricky issue, and kind of a chicken-and-egg problem: given that the information regarding which distribution should be favoured is usually accessible once the depth has been estimated, or the view synthesis problem has been tackled. In a recent work ([93]) focused on monocular view synthesis, the authors tackle this problem of building a non-

uniform MPI distribution for the given scenes. To do so, they assume a depth map of the scene is given as input; analyzing the depth map, they are able to select which depth planes are likely to be the most relevant in the scene. They then only choose the most relevant depth planes (the number of depth planes is set during training), and are able this way to obtain an optimized MPI representation for any given scene.

This process is extremely efficient, and is very convenient for obtaining a compressed and optimized MPI representation for a given scene. Sending as input information a depth map actually gives already a lot of information, that the MPI representation will indeed deepen and improve. In our case, we are interested in evaluating whether it is possible to build a network that can be adjusted to any MPI representation at test time, without requiring ground truth or estimated depth maps. Besides, we also want to be able to build scene representations with a variable number of considered depth planes, during training, as well as at test time.

To address this problem, we draw inspiration from the work in [117], and decide to also resort to a LSTM. The key idea is to associate a LSTM loop with one (α, C) couple of one considered depth plane. Associating a network loop with the processing of a depth plane is indeed very favourable. Changing the number of depth planes to consider between training and test time is extremely easy, given only the number of RNN loops needs to be changed. Besides, changing the depth distribution of the MPI representation is also very accessible, given that it just implies the use of our Recurrent Neural Network on a different sequence of planes.

If this is the guiding factor for choosing RNNs, we are also interested in their lightweightedness; the networks presented in this chapter are around 100,000 parameters, which is a small value by deep learning standards. We also force our network to adjust to several input view configurations during training. This allows us, at test time, to use efficiently several configurations, and it notably gives us the ability to reconstruct a light field using only one network, for a variable number of input views.

5.4.2 Global outline

Following the analysis in the previous section as well as drawing inspiration from the work in [117], we decide to resort to a LSTM architecture. The pipeline is described in

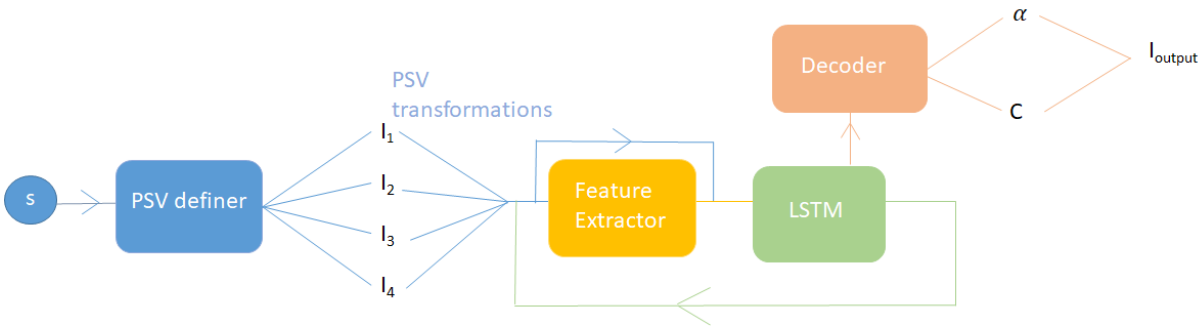

Figure 5.1 – Global outline of our pipeline. An input integer signal s is sent, giving away the requested input configuration and adjusting the PSV transformations to come accordingly (*PSV definer*). If fewer than 4 views are considered, input images and transformations are simply duplicated. We then have 4 images shifted towards the target view with a translation vector depending on the considered depth plane. The features of the 4 shifted images for one specific disparity value are then extracted before being fed, along with the images themselves, to a LSTM. The hidden state cell for every iteration can be decoded using a small neural network (the Decoder), giving us a couple (α, C) accounting for a MPI scene representation. The final image I_{output} is then obtained by combining all α and C.

figure 5.1. First, we take as input 4 views, and we obtain from them a PSV sequence (*Plane-Sweep Volume*). The main idea of PSVs is to shift all the input images towards the target image, by applying sequentially translation vectors in the direction of the target image, with increasing or decreasing vector intensity. Such PSVs are commonly used notably because they represent a process that is computationally not expensive. The main reason, though, is that it is a good way to make sure the network processes the whole sequence as a disparity-variant sequence. We have found experimentally that it was a very efficient way to connect a specific disparity (or depth) value with one specific iteration of our method. Our intuition is that the following network will then produce for every iteration a mask to filter the pixels that should indeed be warped with the disparity value of the current iteration.

We then extract features from the input PSV. To do so, we decide to keep a very simple feature extractor, made up of a succession of several convolutional layers. The main reason behind such a choice is that it allows to keep a small number of parameters throughout the work; besides, not having an encoder-decoder structure with skip-connections for our feature extractor is useful, since it gives fewer constraints for the input image resolution. We augment our feature extractor by adding *Attention-Based modules* for every layer; we

have noted that experimentally, doing so led to an average PSNR increase of **+0.4 dB**, for a negligible number of supplementary parameters. The output of the feature extractor as well as its input PSV are concatenated, and sent as input of a LSTM network. The final image is obtained by decoding the hidden state of the cell, using another very small neural network, the *Decoder*, giving the couples (α, C) associated with the successive depth planes.

The configuration of the 4 input views is randomly selected among a specific number of configurations during training (specified in figure 5.2), and an input signal, different for every configuration, is sent alongside it. This input signal changes the nature of the PSV trajectory to account for the chosen configuration; during training, the objective is also for the network to learn the connection between the value of this input signal and the chosen configuration. At test time, just sending as input the associated signal is enough for choosing a specific configuration and using the proper PSV translation vector accordingly.

5.4.3 Learning metrics

We decide to adopt the following metrics for our predicted view:

$$||I_{target} - \mathcal{O}(\alpha, C)|| \tag{5.1}$$

where I_{target} stands for the target image, and α, C correspond to the MPI representation, and $\mathcal{O}$ being the *over* operator. We thus learn our MPI representation in an *unsupervised* way.

5.4.4 Learning conditions

We train our method on two datasets: one dataset of real-world light fields first presented in [29]. We take the same train-validation-test split as they do. We also evaluate our method by training it and testing it on the synthetic dataset HCI, first presented in [118]. We train our network using TensorFlow ([108]) and Keras ([109]). As the algorithm to optimize our network, we use once more the Adam algorithm ([47]) with the standard set of parameters. We use a learning rate of 0.0001.

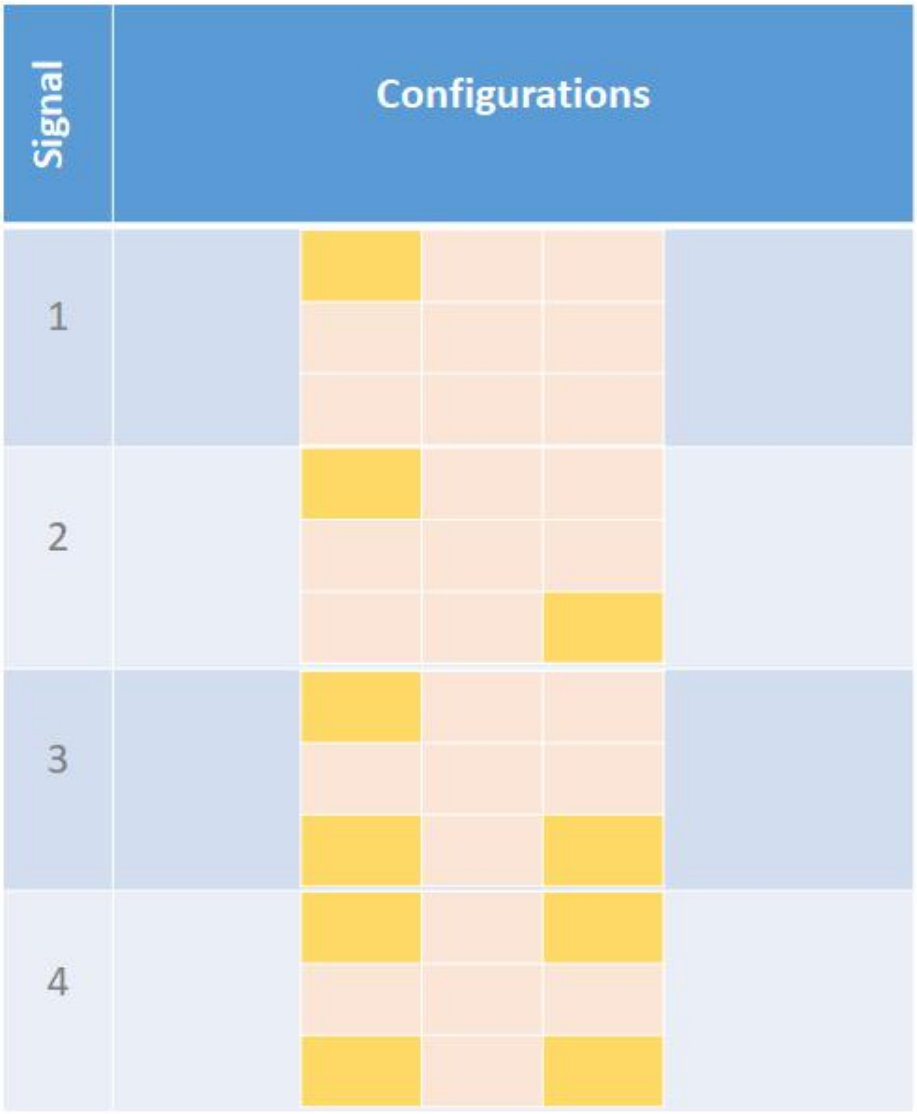

Figure 5.2 – Definition of the various view configurations employed during training. On the left: the input (integer) signal s that is sent, as shown in figure 5.1, on the right: highlighted in yellow, the relative position of the input light field views. We systematically consider corner views in these configurations

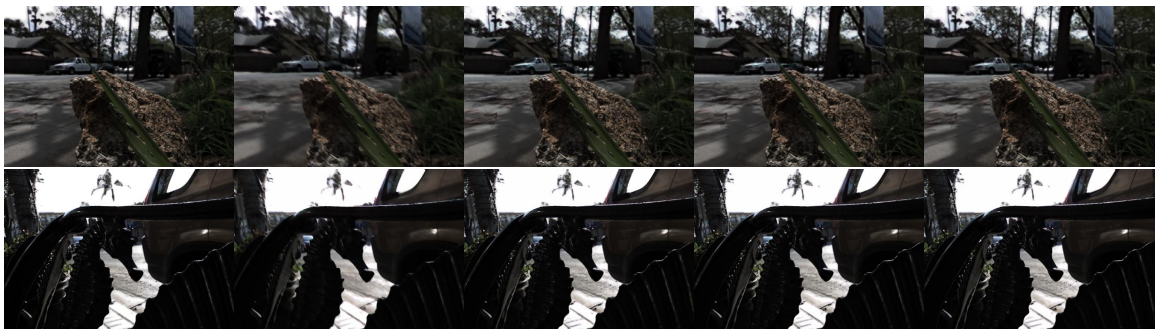

Figure 5.3 – Visual results. From left to right: ground truth image, result of our method for respectively configurations 1, 2, 3 and 4.

Method	Flower1	Cars	Flower2	Rock	Seahorse	Leaves	Average
Ours (4)	**31.87**	**30.78**	**31.32**	**33.16**	**29.94**	27.40	**30.75**
Ours (2)	30.55	29.65	29.17	30.66	28.72	25.64	29.07
[116] (5)	30.00	29.06	28.90	32.60	28.50	**27.74**	29.47

Table 5.1 – Comparing our method with 4 input images, with 2 input images, as well as the method in [116], also for view synthesis using MPI representation.

5.5 Results

We can visually evaluate the performance of our method, for several input views, in figure 5.3. We can also take a look at the visibility map for the successive depth planes, in figure 5.4.

5.5.1 Comparison for MPI representation in a dense setting

We first want to evaluate the performance of our method when compared with another method tackling the problem with the same representation model of MPIs. To do so, we use as a reference the comparison method *Local Light Field Fusion*, first presented in [116]. The method takes 5 input views, and consequently generates a MPI scene representation that will be used for the synthesis of the light field. We train the two methods on the train-validation-test split of the dataset first presented in [29], and we evaluate it in the 4 corner views configuration. For the 5th view, we decide to use the horizontal neighbor to the top-left image. Visual and numerical comparisons are shown in table 5.1 and in figure 5.5.

We can note that our method, on average and in most cases, significantly outperforms

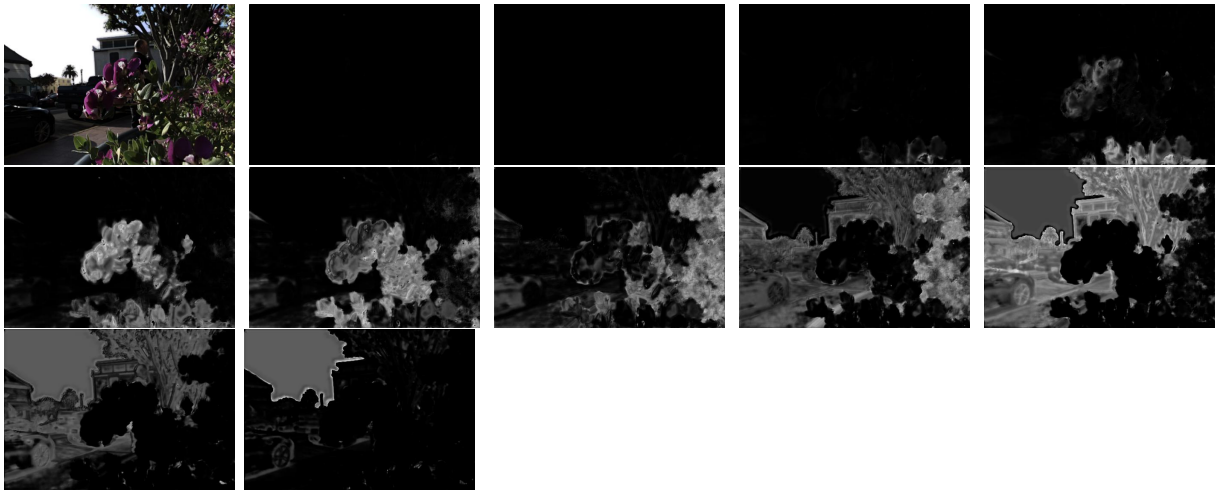

Figure 5.4 – Evaluating the successive visibility masks corresponding to the various depth planes. First: ground truth image, then the visibility masks associated with the successive considered depth planes. The whiter the pixels are, the more they are associated with the given depth plane. We may note a gradual displacement from foreground to background.

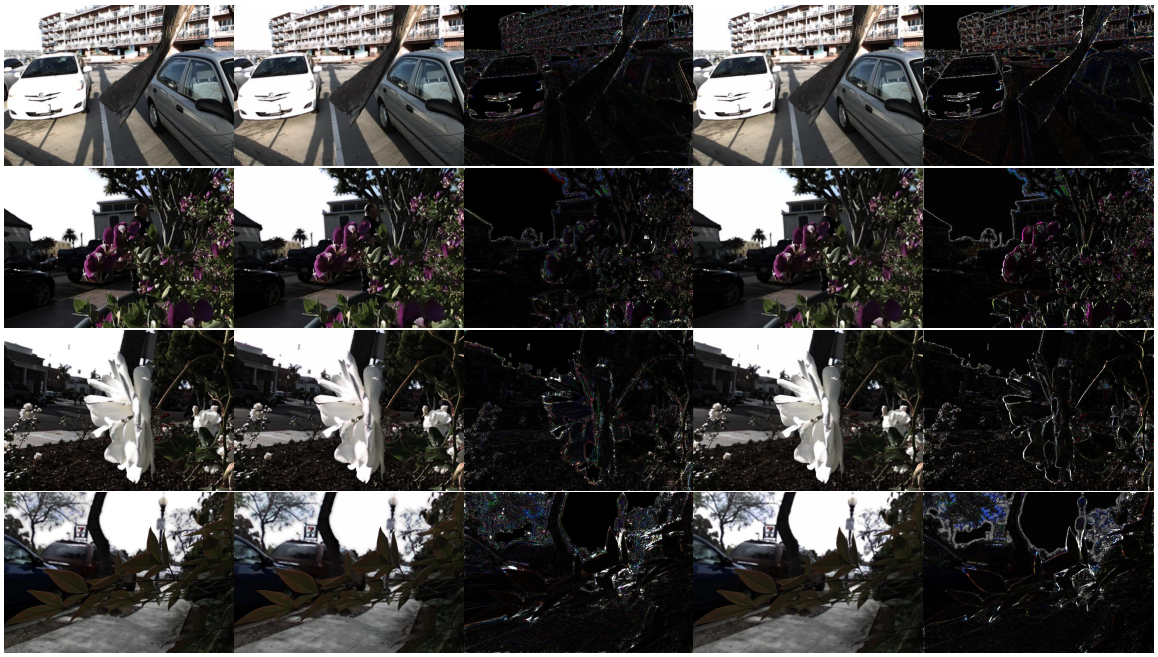

Figure 5.5 – Visual comparison between our method for a 4-view configuration, and the results obtained using [116] on 4 light fields (respectively Cars, Flower1, Flower2, Leaves). From left to right: ground truth image, our prediction, our L1 error, prediction by [116], associated L1 error

the method in [116], both PSNR-wise and visually. We believe that this is an interesting result, for it shows that our method, in spite of its lightweightedness and short training time, is able to improve upon an existing, significant method for light field view synthesis from MPI scene representation. We also evaluate it using a configuration of only 2 input views. We notice that for a certain number of light fields, our method is able to outperform it. On average, [116] performs better, mostly due to Rock and Leaves, two light fields where occluded regions are significant. In this framework, it makes sense that a method with 4 or 5 input views will be able to perform better, given that the supplementary information granted by the other two views is key to the overall performance. We still note rather good-quality results on those light fields even for our 2-view configuration.

From this analysis, we can draw the conclusion that in a dense setting, our process seems to improve upon the MPI representation for light field view synthesis. Of course, state-of-the-art methods go far beyond MPI representations, and we are thus interested in performing, in the next section, a more generic comparison with a variety of methods for light field view synthesis.

5.5.2 General comparison for light field view synthesis

To situate our method within state-of-the-art, we decide to perform comparisons with a range of methods from these past few years that represent as efficiently as possible the various recent trends and models for view synthesis. To do so, we used an evaluation protocol and implementations set up in [72]. Among these other methods, we consider the DIBR-based method [29], with an outline of two successive neural networks, one for depth through DIBR process, and another one for color. Another method we consider is Soft3D, a method for light field view synthesis ([119]) which does not rely on deep learning to perform its visual prediction, but instead analyses the various input images and resorts to jury voting. Another method we compare ourselves with is [71], which performs light field view synthesis by reconstructing epipolar lines using deep learning. Finally, we compare our approach with a more recent method ([72]), performing light field view synthesis by merging two neural networks, one pixel-adapted network as well as one feature-adapted network. Numerical comparisons are shown in table 5.5.2; we give the results obtained while using our 4-view configuration on elements of the test set of [29].

First, we notice that our method outperforms Soft3D, and the epipolar-based methods

Method	Flower1	Cars	Flower2	Rock	Seahorse	Leaves	Average
Ours	31.87	30.78	31.32	33.16	29.94	27.40	30.75
Soft3D	30.29	27.68	30.52	32.67	30.41	27.34	29.82
EPI ([13])	30.44	28.17	29.26	32.46	26.62	26.48	28.90
Kalantari ([29])	33.13	31.53	31.95	34.32	32.03	27.97	31.82
[72]	34.49	32.25	34.19	36.75	34.97	32.53	33.91

Table 5.2 – Comparing our method with reference methods in the field for PSNR values.

for these dense light fields. We believe this is an interesting display of the fact that our method is efficient to process dense light fields. We can also note that performance-wise, our method is clearly outperformed by [72]. This is not particularly surprising, given that the methods described in [72] rely on pre-trained depth estimators, and an optimized pipeline counting millions of parameters, while ours is trained from scratch with only 100,000 parameters.

We also notice that our method does not perform as well as in [29]. The point of the next sections is thus to study what holds back the performance of our approach, and how we can possibly improve it. We may also draw the conclusion that MPI-based methods might not be optimal in a dense context.

5.5.3 Improving the performance

We wonder in this section how we can improve performance, and in particular we try and quantify the loss in performance induced by some of our design choices.

Flexible configuration

In all the results we have shown so far, we have chosen a neural network that was trained on a variety of possible configurations, and not only the evaluated configuration of 4 input corner views. When considering only the 4 corner views as training configuration, we notice an average increase of **+1.23** dB on our result, putting us above [29]'s method for these specific datasets. Hence, the flexible and adaptive framework we have presented seems to reduce the performance of our approach in this specific configuration. It thus seems interesting to study it a bit more to assess and evaluate whether the trade-off is worth it.

Nb img	Flower1	Cars	Flower2	Rock	Seahorse	Leaves
1	19.52/0.73	18.97/0.77	18.96/0.70	18.24/0.66	21.39/0.84	16.51/0.66
2	30.55/0.97	29.65/0.97	29.17/0.96	30.66/0.96	28.72/0.97	25.64/0.93
3	31.71/0.97	30.63/0.98	31.09/0.97	32.23/0.97	29.61/0.98	26.90/0.95
4	**31.87**/0.98	**30.78**/0.98	**31.32**/0.97	**33.16**/0.98	**29.94**/0.98	**27.40**/0.95

Table 5.3 – Evaluating the performance (in PSNR/SSIM) of our method for the 4 configurations presented in figure 5.2.

5.5.4 Analyzing the effect of handling various configurations

We have noticed that alternating configurations leads to a significant decrease in performance when working on the specific evaluation configurations of table 5.5.2. First, we can see a visual and numerical comparison for the various configurations in table 5.5.4 and figure 5.3. We unsurprisingly note that the performance decreases between 2 and 3 input images sent; indeed, when only 2 images are used as input, we lose a lot of information regarding the full light field. We still note that when using only 2 input images, our method remains competitive with other methods presented in table 5.5.2. Unsurprisingly, we also note a sharp decrease in the monocular case, both due to the fact that this is a more complex task, as described in the first two chapters, and this is also related to the fact that this architecture is clearly not a good fit for monocular view synthesis problems.

We see that our method is then able to provide rather satisfying results when only 2 or 3 input images are sent. We know from the analysis in the previous section that this adaptive configuration is detrimental to the 4-view performance. We want to evaluate whether or not this adaptive configuration is beneficial for our 2-view framework. To perform this comparison, we decide to train a neural network only optimized for the 2-view configuration. The comparison is performed in table 5.4.

We want here to study the trade-off that is made. In particular, we are interested in evaluating whether this flexibility improves the result when the number of input views is < 4, in table 5.4. We notice that our Recurrent Neural Network seems to learn from having adaptive configurations during training when only 2 images are sent as input. We can then see that a trade-off has to be found; this adaptive configuration process reduces our performance on 4-view light field view synthesis, but tends to increase the performance that would have been otherwise obtained, while working in a 2-view configuration.

Nb img	Flower1	Cars	Flower2	Rock	Seahorse	Leaves	Average
2 (multi-config network)	**30.55**	**29.65**	**29.17**	30.66	**28.72**	**25.64**	**29.07**
2 (trained on only config 2)	30.00	29.36	29.01	**30.70**	28.39	25.51	28.83

Table 5.4 – Comparison between having 2 input images for a network trained, given several input configurations, and a neural network optimized for this specific 2-image configuration.

Given that the priority we had set while starting the work was to perform efficient view synthesis for a smaller number of input views, we believe the trade-off is worth it, and keep this adaptive configuration process.

The method presented so far has limitations that need to be emphasized. First, one limitation we have already mentioned is its pure performance; the method is not state-of-the-art for light field view synthesis given the 4 corner views. This is unsurprising, given we had set ourselves constraints related to the number of parameters as well as regarding the overall flexibility of the method. This is still a point that needs to be brought forward.

Another issue regarding our method is the extension to sparser contents. Indeed, using our RNN in that framework is likely to cause difficulties; given that we assume one depth plane per RNN loop, if we keep the distribution presented so far of considering planes for every disparity integer value, it will imply many more RNN loops need to be performed for sparse contents, greatly improving the evaluation time. Besides, we also notice our method tends to have a decreasing performance when too many iterations are performed.

5.5.5 Sparse contents

Gradual data learning

We want to evaluate our method on sparser contents, i.e. light fields with a larger baseline. To do so, we decide to train our method on HCI, a synthetic dataset including more examples of sparse contents ([118]). When directly training in the same conditions as in the dense section, we notice that we encounter a lot of difficulties coming up with fully satisfying performance. Thus, we develop a specific training procedure that we call *gradual data learning*. The core idea is to start with simple contents, and a simple task, that we

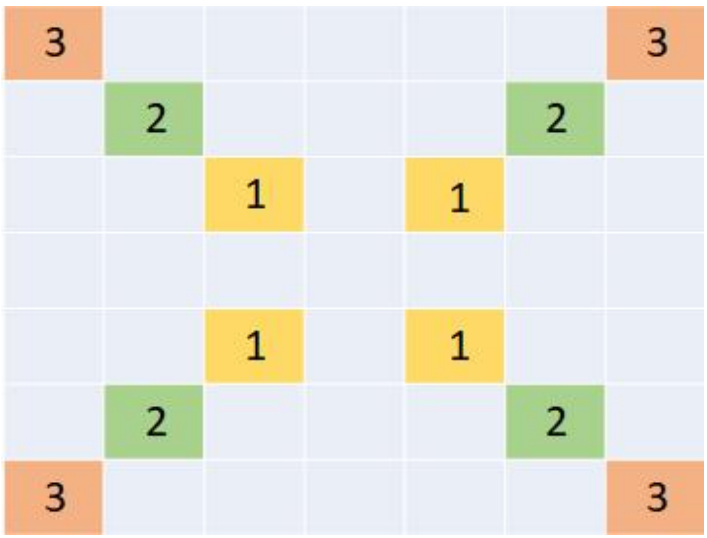

Figure 5.6 – Illustration of our *Gradual Data Learning* process: in a first phase of training, we consider the images labeled '1' as input, we then add the images labeled '2' as possible input, before adding afterwards the images labelled '3'.

gradually extend to more tricky problems as iterations go by. This way, the method first learns how to proceed with a simplified version of the task, before gradually making it more complex. In our case, what we do is that we consider input views that are further and further away from the central view, starting with the 4 views around the central view. Then, we gradually add as possible input configurations with views that are further away from the central view. The process is described in figure 5.6. We have also found experimentally that proceeding this way allowed us to sharply reduce the training time. Using this procedure allows us to move from dense datasets to sparse datasets and proved to be an efficient way to tackle this problem.

Visual evaluation

Visual results are shown in figure 5.7. We notice that we obtain overall a final prediction with good visual quality.

The problem of the computational time

If the method can produce adequate results in a sparse context, it was not built with this purpose in mind. In particular, when adapting to sparse contents, given that one LSTM loop is associated with one possible disparity unit, more sparsity comes with more

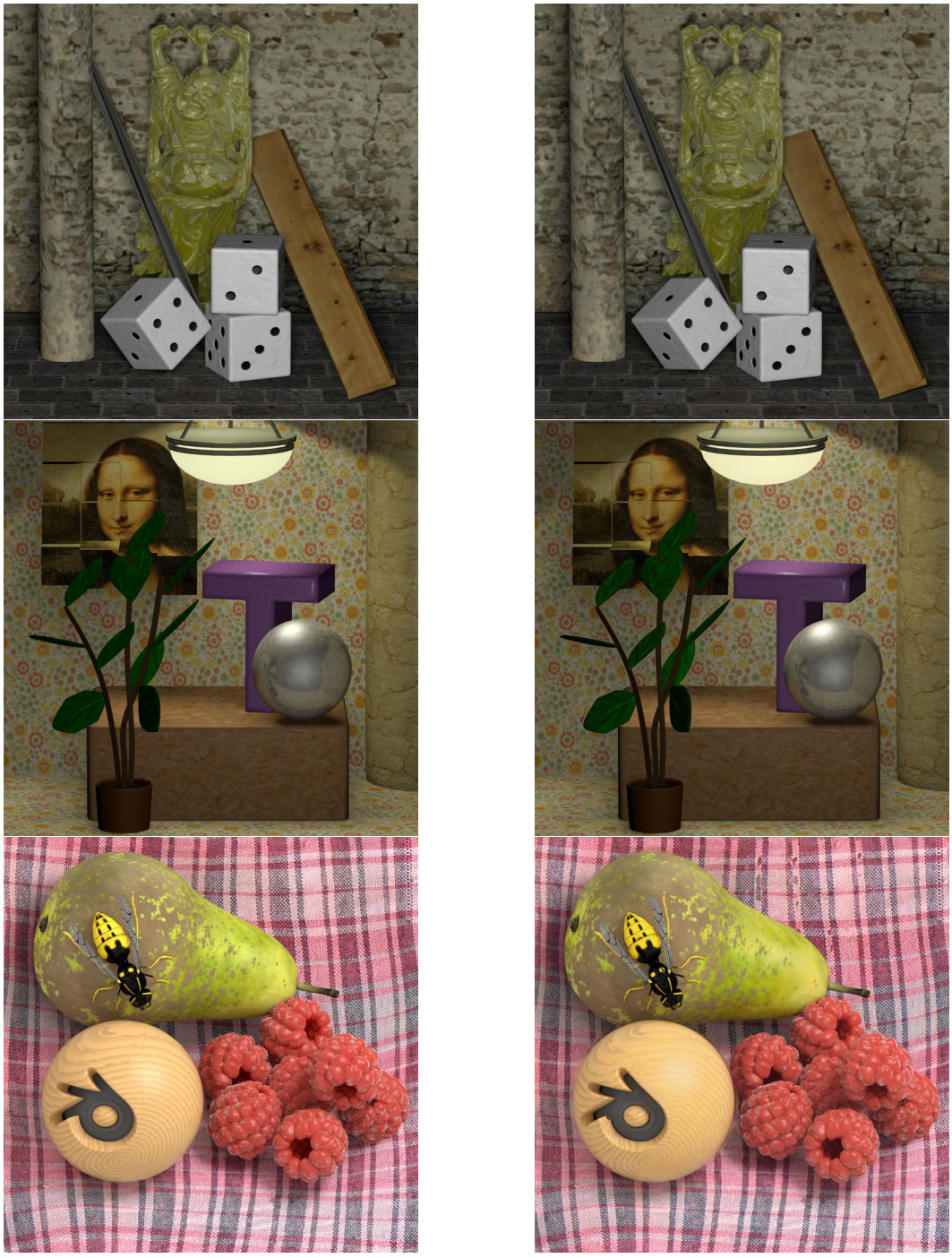

Figure 5.7 – Examples of visual results on the sparse light field dataset HCI for 4 input views. From left to right: ground truth image, our prediction. PSNR for Buddha: **40.01**, PSNR for Mona: **38.87**, PSNR for StillLife: **29.81**.

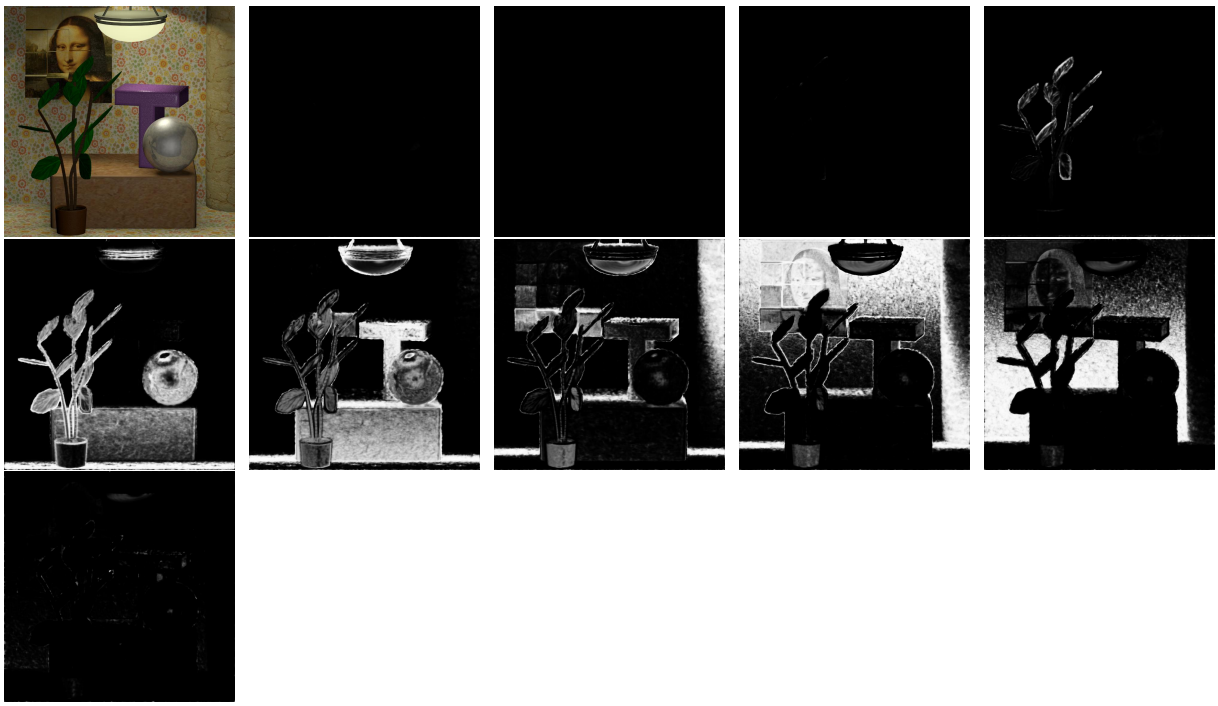

Figure 5.8 – Evaluating the successive visibility masks corresponding to the various depth planes. First: ground truth image, then the visibility masks associated with the successive considered depth planes. The whiter the pixels are, the more they are associated with the given depth plane. We may note a gradual displacement from foreground to background.

RNN loops, and thus a longer computational time.

This is where the LSTM structure of our method proves convenient, though. Given that the MPI structure of our considered scene is not set between training and testing, we can modify it at will while keeping the efficiency of our method. When working on sparse contents, it means that we can produce at will MPI representations that will correspond to a selection of specific depth planes, and use our method in this representation. This is an efficient way to reduce the computational burden in this situation of sparsity. An example of MPI analysis from the Mona light field from HCI is displayed in figure 5.8.

5.6 Conclusion and prospects

In this chapter, we have presented a method performing the task of light field view synthesis given a subset of input images (between 2 and 4). The method uses a MPI-based

structure to reconstruct the full scene. By using a Recurrent Neural Network architecture, and linking one RNN loop to a specific depth plane, we can have a method that performs well in any chosen distribution of depth planes, thus extending MPIs to new interesting distributions. Besides, the method can work from a variable number of input images; even though it is not state-of-the-art when considering a common configuration with other methods, it has good performance, especially regarding the overall lightweightedness and the flexibility of the method.

Most interesting developments would be related to pushing forward the idea of having a chosen distribution of depth planes in the MPI representation model. We are faced with one problem, still: we have not managed to make this process of analyzing the most relevant depth planes an automatic one. In [93], this process is made automatic by a prior depth map estimation. We also studied leads in this thesis to try and make the process automatic without resorting to more ground truth (or pre-trained) elements. One idea we had was to use a Attention-Based Module in the decoder, so as to make it choose automatically the most relevant depth planes for MPI estimation. If using a Confidence-Based block in the Feature extractor proved to be efficient, this particular use of Confidence Block did not return the expected results. Another idea that could be envisioned is to use a succession of two Recurrent Neural Networks: one having the role described in the chapter, and another one suggesting possible depth distributions. If the idea was appealing, we found its implementation difficult, practically. Finally, another idea interesting to envision could be to use this kind of architecture and model in an *internal learning* framework ([120]), so as to automatically learn from data the right distribution of depth planes in this configuration.

Chapter 6

General conclusion

6.1 Summary

In this thesis, we addressed the problem of view synthesis in a light field context, in particular for a very small number of input views (≤ 4). We first considered the problem of light field monocular view synthesis: how can we generate a light field from a single image?

To address this problem, we have first decided to focus on a stereo case framework, to test our concepts. We proposed in **chapter 3** a pipeline to generate a new range of horizontal views from one single image. Besides, our method is able to estimate a depth map and to estimate a confidence measure in its own prediction. To do so, we have designed a pipeline to perform the synthesis of these new views: it is based on the combination of several types of prediction. One depth-based prediction, with a first neural network (the *Disparity-Based Predictor*) that aims at estimating the depth of the given scene, and which performs view synthesis by warping. The predictions obtained this way are necessarily limited with the fact that occluded regions are not explicitly handled, and with the fact that a small error in the prediction of the depth map can lead to significant mistakes for the image finally produced. To address these concerns, a second neural network (the *Confidence-Based Merger*) is tasked with identifying such regions. Finally, a third neural network (the *Refiner*) focuses on these regions to improve their outlook. The entire pipeline allows to have both a well-estimated depth map and a self-confidence measure, besides the views finally produced. The method outperformed state-of-the-art approaches at the time of submitting.

Chapter 4 then extends this work to light fields, and to improve some of its key elements. Extension to light fields can seem rather straightforward, but actually comes with significant obstacles. First, monocular view synthesis methods are usually trained from datasets with similar semantics and geometrical configurations, which are harder

to capture in a light field framework. For that reason, we design a stereo-to-light fields module allowing to also train our method on stereo contents, and to apply our Refiner on other views of the light field. Besides, we also bring significant improvements to our method, by adding an adversarial component to our Refiner, modifying and improving the Confidence Measure estimation process, to make it more flexible. Our method is able to outperform reference methods (such as [90]) in the field on the Flowers dataset. We also validate our stereo-to-light-fields module by evaluating it in the same framework, and notice it is efficient. We also check the nature of the light fields views we produce from the stereo dataset KITTI.

Finally, in **chapter 5**, we relax the constraint by considering several possible input views (up to 4 views). Given we have access to more information, and given we want to keep our constraint of having as few parameters as possible, we decide this time to resort to a LSTM architecture, inspired by [18]. We estimate this way a MPI representation for a given scene, and we have an extremely lightweight neural network (around 100,000 parametes) with good performance, which is able to work from a variable number of input images without retraining.

6.2 Future work

We believe that the work carried out in this thesis opens interesting research directions.

The field of deep learning for monocular view synthesis was relatively nascent when we started working on the subject, and since then, many new works have been presented, deeply changing the current landscape of the field.

A salient problem in monocular view synthesis that seems to overshadow all the other issues is the matter of the semantic genericity of the presented methods. Indeed, all the monocular methods presented in this report can only work for a very specific semantics and geometrical configuration and usually fail when a drastic change happens between the training and the test set. The next logical question to ask is then: how can we mitigate this lack of genericity? We have tried in our own work to answer this question by bringing up a method able to adjust quickly and with a relatively low number of input views for re-training. We still wonder whether more could be done to improve this genericity. To

address this problem, we see two main leads.

The first lead was mentioned in **chapter 4**. Even though our own efforts for this matter have not proven to give results that were consistently good enough, we still believe that resorting to a proxy task could somehow be a valid way forward. Should we be able to inject the results obtained from a proxy task from one single image (such as segmentation for instance) on specific datasets, and use them to guide our networks (drawing inspiration from [114]), we would be likely to obtain a better genericity. After exploring this lead, we had difficulties finding ways to improve our own predictions using some proxy tasks. Finding a way to use guidance networks to allow monocular methods to work for more generic data would be an interesting way forward, given it would allow a much more significant number of datasets to be used for this task.

The other solution is to focus on data. Today, there are not that many light field datasets that can be efficiently used for training our methods. KITTI is a very good example that was widely used in this thesis, but it is bound to urban scenes with a specific geometrical configuration. Flowers displays scenes that might be now a bit too simple, semantic-wise and from a geometrical perspective, to fully evaluate and validate the various methods. One idea would be to build a series of light field datasets corresponding to several configurations and specific types of scenes. Training monocular approaches on these series of datasets could be an efficient way to improve the genericity of such methods. We turned around this idea throughout the thesis. In early 2020, we had envisioned to capture this light field dataset using a Lytro camera; the context of the pandemic made this capture process difficult. In spite of not being able to carry it out during these 3 years, we absolutely believe that it is an extremely interesting way forward.

In the best-case scenario, we hope that a single neural network could in a straightforward way be trained and learn from this variety of data, so as to be able to process different scenes and configurations. In case this might prove to be difficult, we believe that the idea developed in **chapter 5**, with a Neural Network capable of adjusting to various view configurations and input signals could also be used. We can imagine a method that can be adjusted to different modes and semantics just by sending the right input signal for instance.

Another way to proceed could be to use other data types that are not light fields. We started exploring that route a bit by using stereo datasets in **chapter 4**. Ideas revolved around using YouTube videos as multi-view captures instead of light fields, or images from video games. All our methods learn the scope and the range of light field and stereo data in a fully implicit way, meaning that our training set needs to have been captured using the same baseline. This, of course, restricts the nature of the data that can be used as input, and makes it difficult to use raw images from these sources in our work. We still believe that this process is also key to making our approach more generic.

To make our approach trainable on various baselines, one idea widely used in the literature ([63]) could be to resort to pose estimation, so as to generate the new view positions directly from vector coordinates. We believe that this is interesting. But we would also like to point out that our own experience working with such methods has shown that it is very prone to small errors in pose estimation methods, which is common, given that automatic pose estimation is a research problem in itself. We thus believe that making the step from implicit pose estimation to automatic pose estimation is not straightforward, and should only be pursued when it is necessary.

We also believe that the complexity of the monocular network (around 6M parameters) could be reduced even more. The experiments we carried out using network compression methods seemed to indicate that they could be efficient ways to reduce even more the parameter burden, all while keeping a similar efficiency. Finally, we believe that a very interesting development for monocular view synthesis would also be to build standards for comparison and evaluation. As of today, the evaluation modalities, as well as the training-validation-test splits used in different articles are usually different. If of course, within the same article, the same evaluation modalities are employed, it still entails that it is very difficult to evaluate and have a fair comparison ground between methods from different articles. We believe a valid way forward could be to build a clear procedure for evaluation, possibly using a particularly relevant and useful dataset, which would allow easy and fair comparisons between the various state-of-the-art methods.

Regarding **chapter 5**, we believe that more significant developments could be pursued. Notably, the approach right now is optimized for dense light fields, and encounters performance issues when working with sparser contents. Promising ideas revolved around the

resort to Attention-Based Modules for choosing the right loops to process. The method is also not suited for monocular view analysis, and it would be interesting to study whether there are ways to improve its results for this particular problem.

As a conclusion, we would like to emphasize that in this thesis, we have tried to tackle the problem of light field view synthesis in a situation where too few views were available for full and faithful reconstruction. If a lot of work still needs to be done to build entirely satisfying pipelines for this task, we believe that this thesis was a first, interesting step into using deep learning for this challenging and ill-posed task.

PUBLICATIONS

[1] S. Evain and C. Guillemot. « A Lightweight Neural Network for Monocular View Generation with Occlusion Handling ». In: *PAMI* (2020).

[2] S. Evain and C. Guillemot. « A Neural Network with Adversarial Loss for Light Field View Synthesis from a Single Image ». In: *VISAPP* (2021).

BIBLIOGRAPHY

[1] E.H. Adelson and J.R. Bergen. « The Plenoptic Function and the Elements of Early Vision ». In: *Computation Models of Visual Processing, M.Landy and J.A. Movshon, eds., MIT Press, Cambridge* (1991).

[2] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. « The Lumigraph ». In: *23rd Annual Conf. on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. 1996, pp. 43–54.

[3] B. Wilburn, N. Joshi, V. Vaish, E-V. Talvala, E. Antunez, a. Barth, A. Adams, M. Horowitz, and M. Levoy. « High Performance Imaging Using Large Camera Arrays ». In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 765–776. ISSN: 0730-0301.

[4] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. *Light Field Photography with a Hand-Held Plenoptic Camera*. Tech. rep. CSTR 2005-02, Apr. 2005.

[5] T. Adelson and J. Wang. « Single Lens Stereo with a Plenoptic Camera ». In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 99–106.

[6] C. Guillemot and R. Farrugia. « Light field image processing: overview and research issues ». In: *IEEE COMSOC MMTC Communications - Frontiers* (2017).

[7] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz. « High-Speed Videography Using a Dense Camera Array ». In: *CVPR* 2.2 (2004), pp. 294–301.

[8] J.C. Yang. « A light field camera for image based rendering ». In: *PhD thesis*. 2000.

[9] H.-N. Nguyen and C. Guillemot. « Color and angular reconstruction of light fields from incomplete-color coded projections ». In: *ICASSP* (2020).

[10] R. Ng. « Digital Light Field Photography ». In: (2006).

[11] J. Shi, X. Jiang, and C. Guillemot. « A framework for learning depth from a flexible subset of dense and sparse light field views ». In: *IEEE Transactions on Image Processing* (2019).

[12] S. Heber, W. Yu, and T. Pock. « Neural EPI-volume networks for shape from light field ». In: *IEEE International Conference on Computer Vision (ICCV)* (2017).

[13] C. Shin, Y. Jeon H.-G.and Yoon, I. S. Kweon, and S.J. Kim. « EPINET: A fully-convolutional neural network using epipolar geometry for depth from light field images ». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[14] X. Jiang, J. Shi, and C. Guillemot. « Depth estimation with occlusion handling from a sparse set of light fields ». In: *IEEE International Conference on Image Processing (ICIP)* (2018).

[15] Ren Ng. « Fourier Slice Photography ». In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 735–744.

[16] K. Yucer, A. Sorkine-Hornung, O. Wang, and O. Sorkine-Hornung. « Efficient 3D object segmentation from densely sampled light fields with applications to 3d reconstruction ». In: *ACM Transactions on Graphics* 35 (2016), pp. 249–257.

[17] Q. Z. H. Zhu and Q. Wang. « 4D light field super-pixel and segmentation ». In: *IEEE Computer Vision and Pattern Recognition* (2017).

[18] M. Hog, N. Sabater, and C. Guillemot. « Light field segmentation using a ray-based graph structure ». In: *IEEE European Conference on Computer Vision (ECCV)* (2016).

[19] F. L. Zhang, J. Wang, E. Shechrman, Z. Y. Zhou, J. X. Shi, and S. M. Hu. « Plenopatch: Patch-based plenoptic image manipulation ». In: *IEEE Transactions on Visualization and Computer Graphics* (2016).

[20] O. Frigo and C. Guillemot. « Epipolar Plane Diffusion: An Efficient Approach for Light Field Editing ». In: *British Machine Vision Conference (BMVC)* (2017).

[21] C. Galdi, V. Chiesa, C. Busch, P. Correia, and J.-L. Dugelay. « Light Fields for Face Analysis ». In: *MDPI* (2019).

[22] M. Liu, H. Fo, Y. Wei, Y. Rehman, L. Po, and W. Lo. « Light field-based face liveness detection with convolutional neural networks ». In: *SPIE Electron. Imaging* (2019).

[23] A. Sepas-Moghaddam, M.A. Haque, P.L. Correia, K. Nasrollahi, T.B. Moeslund, and F.A. Pereira. « Double-Deep Spatio-Angular Learning Framework for Light Field based Face Recognition ». In: *IEEE Transactions on Circuits Syst. Video Technol.* (2019).

[24] V. Chiesa and J.L. Dugelay. « On Multi$_V iew Face Recognition Using Lytro Images$ ». In: *EUSIPCO* (2018).

[25] F. Hawary, G. Boisson, C. Guillemot, and P. Guillotel. « Compressive 4D Light Field Reconstruction Using Orthogonal Frequency Selection ». In: *International Conference on Image Processing (ICIP)* (2018).

[26] M. Gupta, A. Jauhari, K. Kulkarni, S. Jayasuriya, A. Molnar, and P. Turaga. « Compressive Light Field Reconstruction using Deep Learning ». In: *Workshop of the IEEE Computer Vision and Pattern Recognition Conference (CVPRW)* (2017).

[27] X. Jiang, M. Le Pendu, and C. Guillemot. « Light field compression using depth image based view synthesis ». In: *IEEE International Conference on Multimedia and Expo Workshops* (2017).

[28] Y. Yoon, H.-G. Jeon, D. Yoo, J.-Y. Lee, and I.S. Kweon. « Liht-Field Image Super-Resolution using Convolutional Neural Networks ». In: *IEEE Signal Processing Letters* (2017).

[29] Kalantari N., T. Wang, and R. Ramamoorthi. « Learning-based view synthesis for light field cameras ». In: *SIGASIA* (2016).

[30] R. Farrugia, C. Galea, and C. Guillemot. « Super-Resolution of Light Field Images using Convolutional Neural Networks ». In: *IEEE Journal on Selected Topics in Signal Processing* (2017).

[31] R. Farrugia and C. Guillemot. « Light Field SUper-Resolution using a Low-Rank Prior and Deep Convolutional Neural Networks ». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).

[32] C. Fehn. « A 3D-TV approach using Depth-Image-Based Rendering ». In: *Proceedings of 3rd IASTED Conference on Visualization, Imaging, and Image Processing, Benalmadena* (2003).

[33] C. Fehn. « Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV ». In: *Proc. SPIE 5291, Stereoscopic Displays and Virtual Reality Systems XI* (2004).

[34] M. Eisemann, B.D. Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. « Floating Textures ». In: *Floating textures* (2008).

[35] M. Goesele, J. Ackermann, S. Furhmann, C. Haubold, and R. Klowsky. « Ambient point clouds for view interpolation ». In: *ACM Transactions on Graphics* (2010).

[36] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. « Depth Synthesis and Local Warps for Plausible Image-based Navigation ». In: *ACM Transactions on Graphics* (2013).

[37] S. Wanner and B. Goldluecke. « Globally Consistent Depth Labeling of 4D Light Fields ». In: *CVPR* (2012).

[38] S. Wanner and B. Goldluecke. « Variational Light Field Analysis for Disparity Estimation and Super-Resolution ». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014).

[39] M.W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. « Depth from combining defocus and correspondence using light-field cameras ». In: *ICCV* (2013).

[40] T.C. Wang, A.A. Efros, and R. Ramamoorthi. « Occlusion-aware depth estimation using light-field cameras ». In: *ICCV* (2015).

[41] E. Penner and L. Zhang. « Soft 3D Reconstruction for View Synthesis ». In: *ACM Transactions on Graphics* (2017).

[42] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand. « Light field reconstruction using sparsity in the continuous Fourier domain ». In: *ACM Transactions on Graphics* (2014).

[43] F. Hawary, C. Guillemot, D. Thoreau, and G. Boisson. « Scalable Light Field Compression Scheme using Sparse Recognition and Restoration ». In: *International Conference on Image Processing (ICIP)* (2017).

[44] S. Vagharshakyan, R. Bregovic, and A. Gotchev. « Light field reconstruction using shearlet transform ». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015).

[45] M. Le Pendu, C. Guillemot, and A. Smolic. « A Fourier Disparity Layer Representation for Light Fields ». In: *IEEE Transactions on Image Processing (TIP)* (2019).

[46] F. Rosenblatt. « The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain ». In: *Psychological Review* (1958).

[47] D.P. Kingma and J. Ba. « Adam: a method for stochastic optimization ». In: *International Conference on Learning Representations (ICLR)* (2015).

[48] S. Ruder. « An overview of gradient descent optimization algorithms ». In: *Technical report* (2016).

[49] L. Sifre. « Rigid-motion scattering for image classification ». In: *PhD thesis* (2014).

[50] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreeto, and H. Adam. « Mobilenets: Efficient convolutional neural networks for mobile vision applications ». In: *arXiv* (2017).

[51] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. « Generative Adversarial Networks ». In: *NIPS* (June 2014).

[52] M. Arjovsky, S. Chintala, and L. Bottou. « Wasserstein GANs ». In: *ICML* (2017).

[53] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. « Spectral Normalization for Generative Adversarial Networks ». In: *ICPR* (2018).

[54] S. Hochreiter and J. Schmidhuber. « Long short-term memory ». In: *Neural Computation* (1997).

[55] K. Simonyan and A. Zisserman. « Very deep convolutional networks for large-scale image recognition ». In: *International Conference on Learning Representation (ICLR)* (2015).

[56] A. Krizhevsky, I. Sutskever, and G. Hinton. « ImageNet Classification with Deep Convolutional Neural Networks ». In: *ACM Transactions on Graphics* (2012).

[57] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. « MobileNetV2: Inverted Residuals and Linear Bottlenecks ». In: *CVPR* (2018).

[58] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. « Spatial Transformer Networks ». In: *NIPS* (2015).

[59] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. « Video frame synthesis using deep voxel flow ». In: *Proceedings of International Conference of Computer Vision* (2017).

[60] J. von Amersfoort, W. Shi, A. Acosta, F. Massa, J. Totz, Z. Wang, and J. Caballero. « Frame Interpolation with Multi-Scale Deep Loss Functions and Generative Adversarial Networks ». In: *arXiv* (2017).

[61] X. Cun, F. Xu, C. Pun, and H. Gao. « Depth Assisted Full Resolution Network for Single Image-based View Synthesis ». In: *CoRR* (2017).

[62] T. Zhou, M. Brown, N. Snavely, and D. Lowe. « Unsupervised Learning of Depth and Ego-Motion From Video ». In: *CVPR* (2017).

[63] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. « Stereo Magnification: Learning View Synthesis using Multiplane Images ». In: *SIGGRAPH* (2018).

[64] B. Mildenhall, P. Srinivasan, R. Ortiz-Cayon, N. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. « Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines ». In: *ACM Transactions on Graphics* (2019).

[65] R. Tucker and N. Snavely. « Single-view View Synthesis with Multiplane Images ». In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[66] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q.V. Le. « Attention Augmented Convolutional Networks ». In: *ICCV* (2019).

[67] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao. « YOLOv4: Optimal Speed and Accuracy of Object Detection ». In: *arXiv* (2020).

[68] M.S. Khan Gul, U. Mukati, M. Batz, S. Forchhammer, and J. Keinert. « Light-field view synthesis using convolutional block attention module ». In: *arXiv* (2020).

[69] H. Ling, J. Wu, J. Huang, J. Chen, and P. Li. « Attention-based convolutional neural network for deep face recognition ». In: *Multimedia Tools and Aplications* (2020).

[70] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. « CBAM: Convolutional Block Attention Module ». In: *ECCV* (2018).

[71] G. Wu, Y. Liu, Q. Dai, and T. Chai. « Learning sheared EPI structure for light field reconstruction ». In: *IEEE Transactions on Image Processing* (2019).

[72] J. Shi, X. Jiang, and C. Guillemot. « Learning Fused Pixel and Feature-based View Reconstructions for Light Fields ». In: *CVPR* (2020).

[73] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: `2003.08934 [cs.CV]`.

[74] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. « Context encoders: Feature Learning by Inpainting ». In: *CVPR* (2016).

[75] A. Clark, J. Donahue, and Simonian K. *Adversarial Video Generation on Complex Datasets*. 2019. arXiv: 1907.06571 [cs.CV].

[76] A. Saxena, M. Sun, and A. Ng. « Make3D: Learning 3D Scene Structure from a Single Still Image ». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009).

[77] D. Eigen, C. Puhrsch, and R. Fergus. « Depth map prediction from a single image using a multi-scale deep network ». In: *NIPS* (2014).

[78] Y. Cao, Z. Wu, and C. Shen. « Estimating depth from monocular images as classification using deep fully convolutional residual networks ». In: *IEEE Transactions on Circuits and Systems for Video Technology* (2016).

[79] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. « Deeper depth prediction with fully convolutional residual networks ». In: *3DV* (2016).

[80] M. Pinheiro de Carvalho, F. Champagnard, and A. Almansa. « Deep Depth from Defocus: Neural Networks for Monocular Depth Estimation ». In: *PhD thesis* (2019).

[81] C. Godard, O. Mac Aodha, and G. Brostow. « Unsupervised Monocular Depth Estimation with Left-Right Consistency ». In: *CVPR* (2017).

[82] Y. Kuznietsov, J. Stückler, and B. Leibe. « Semi-Supervised Deep Learning for Monocular Depth Map Prediction ». In: *CVPR* (2017).

[83] Y. Horry, K. Anjyo, and K. Arai. « Tour into the picture: using a spidery mesh interface to make animation from a single image ». In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 225–232.

[84] D. Hoiem, A. Efros, and M. Hebert. « Automatic photo pop-up ». In: *ACM Transactions on Graphics* (2005).

[85] T. Kulkarni, W. Whitney, P. Kohli, and J. Tenebaum. « Deep convolutional inverse graphics network ». In: *Advances in Neural Information Processing Systems* (2015).

[86] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. Efros. « View synthesis by appearance flow ». In: *ECCV* (2016).

[87] S. Sun, M. Huh, Y. Liao, N. Zhang, and J. Lim. « Multi-view to Novel view: Synthesizing novel views with Self-Learned Confidence ». In: *ECCV* (2018).

[88] M. Liu, X. He, and M. Salzmann. « Geometry-aware Deep Network for Single-Image Novel View Synthesis ». In: *arXiv preprint* (2018).

[89] J. Xie, R. Girshick, and A. Farhadi. « Deep3D: Fully Automatic 2D-to-3D Video Conversion with Deep Convolutional Neural Networks ». In: *ECCV* (2016).

[90] P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, and R. Ng. « Learning to Synthesize a 4D RGBD Light Field from a Single Image ». In: *ICCV* (2017).

[91] A. Ivan, Williem, and I. Kyu Park. *Synthesizing a 4D Spatio-Angular Consistent Light Field from a Single Image*. 2019. arXiv: 1903.12364 [cs.CV].

[92] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. « 3D Photography using Context-aware Layered Depth Inpainting ». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[93] Q. Li and N.K. Kalantari. « Synthesizing Light Field From a Single Image with Variable MPI and Two Network Fusion ». In: *ACM Transactions on Graphics* (2020).

[94] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. Berg. « Transformation-Grounded Image Generation Network for Novel 3D View Synthesis ». In: *CVPR* (2017).

[95] S. Tulsiani, R. Tucker, and N. Snavely. « Layer-structured 3D Scene Inference via View Synthesis ». In: *ECCV* (2018).

[96] Lingyan Ruan, Bin Chen, and Miu Ling Lam. « Light Field Synthesis from a Single Image using Improved Wasserstein Generative Adversarial Network ». In: *EG 2018 - Posters*. Ed. by Eakta Jain and Jirí Kosinka. The Eurographics Association, 2018. DOI: 10.2312/egp.20181017.

[97] O. Woodford, I. Reid, P. Torr, and A. Fitzgibbon. « On new view synthesis using multiview stereo ». In: *BMVC* (2007), pp. 1–10.

[98] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. « Deepstereo: Learning to predict new views from the world's imagery ». In: *CoRR* (2015).

[99] T. Habtegebrial, K. Varanasi, C. Bailer, and D. Stricker. « Fast View Synthesis with Deep Stereo Vision ». In: *arXiv preprint* (2018).

[100] A. Geiger, P. Lenz, and R. Urtasun. « Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite ». In: *CVPR* (2012).

[101] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. « Vision meets Robotics: The KITTI Dataset ». In: *International Journal of Robotics Research* (2013).

[102] M. Menze and A. Geiger. « Object Scene Flow for Autonomous Vehicles ». In: *CVPR* (2015).

[103] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. « Image Quality Assessment: From Error Visibility to Structural Similarity ». In: *IEEE Transactions on Image Processing* (2004).

[104] R. Zhang, P. Isola, A. Efros, E. Shechtman, and O. Wang. « The Unreasonable Effectiveness of Deep Features as Perceptual Metric ». In: *CVPR* (2018).

[105] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. « ImageNet: A large-scale hierarchical image database ». In: *CVPR* (2009).

[106] M. Mathieu, C. Couprie, and Y. LeCun. « Deep multi-scale video prediction beyond mean square error ». In: *International Conference on Learning Representations* (2016).

[107] H. Zimmer, A. Bruhn, and J. Weickert. « Optic flow in harmony ». In: *IJCV* (2011).

[108] Martı'n Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[109] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[110] K. Yamaguchi, D. McAllester, and R. Urtasun. « Efficient joint segmentation, occlusion labeling, stereo and flow estimation ». In: *ECCV* (2014).

[111] N. Mayer, E. Ilg, P. Häusser, and P. Fischer. « A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow and Scene Flow Estimation ». In: *CVPR* (2016).

[112] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. « The Cityscapes Dataset for Semantic Urban Scene Understanding ». In: *CVPR* (2016).

[113] S. Hadfield and R. Bowden. « Hollywood 3D: Recognizing Actions in 3D Natural Scenes ». In: *CVPR* (2013).

[114] V. Guizilini, R. Hou, J. Li, R. Ambrus, and A. Gaidon. « Semantically-Guided Representation Learning for Self-Supervised Monocular Depth ». In: *ICLR* (2020).

[115] S. Evain and C. Guillemot. « A Lightweight Neural Network for Monocular View Generation with Occlusion Handling ». In: *PAMI* (2020).

[116] B. Mildenhall, P.P. Srinivasan, R. Ortiz-Cayon, N. Khademi Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. « Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines ». In: *ACM Trans. on Graphics (TOG)* (2019).

[117] M. Hog, N. Sabater, and C. Guillemot. « Long Short Term Memory Networks For LightField View Synthesis ». In: *IEEE International Conf. on Image Processing, ICIP*. Sept. 2019.

[118] S. Wanner, S. Meister, and B. Goldluecke. « Datasets and benchmarks for densely sampled 4D light fields ». In: *Conference on Vision, Modelling and Visualization* (2013).

[119] E. Penner and L. Zhang. « Soft 3D Reconstruction for View Synthesis ». In: *ACM Trans. on Graphics (SIGGRAPH Asia)* 36.6 (2017).

[120] D. Ulyanov, A. Vedaldi, and V. Lempitsky. « Deep Image Prior ». In: *CVPR* (2018).

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Apprentissage profond pour la synthèse de vues de champs de lumière dans un cadre monoculaire et pour un faible nombre de vues d'entrée

Mot clés : champs de lumière, synthèse de vues, génération de vues monoculaire, apprentissage profond

Résumé : Un champ de lumière est une modélisation de l'intensité de l'ensemble des rayons lumineux circulant par tous les points d'une scène donnée. Capturer de tels champs de lumière en utilisant des appareils adéquats est intéressant, car permet d'importants développements et applications. Cependant, ces champs de lumière sont très coûteux en stockage données. Nous répondons à cette problématique dans cette thèse par le biais de la synthèse de vues. Nous présentons ainsi trois contributions. Dans un premier temps, nous nous intéressons à la synthèse de nouvelles vues à partir d'une seule image. Nous nous focalisons en particulier sur le cas stéréoscopique, et par le biais d'une combinaison avantageuse de plusieurs réseaux de neurones, nous présentons une méthode se comparant favorablement avec l'état de l'art, tout en étant très légère et capable de s'ajuster rapidement à de nouveaux jeux de données. Dans un second temps, nous étendons cette contribution au bidimensionnel, en générant cette fois des champs de lumière. Nous améliorons également le traitement de la problématique en ajoutant un composant adversarial, et développons un module stéréo-champ de lumière permettant d'entraîner notre approche sur des données stéréoscopiques tout en générant des champs de lumière de qualité. Enfin, dans un dernier temps, nous utilisons des réseaux de neurones récurrents pour générer des champs de lumière entiers à partir d'un nombre libre de vues d'entrée, en adoptant une représentation dite "MPI". Notre méthode est légère, efficace et peut s'appliquer dans n'importe quelle distribution de plans de profondeur lors du test.

Title: Deep learning for light field view synthesis from monocular and a very sparse set of input views

Keywords: light fields, view synthesis, monocular view generation, deep learning

Abstract: A light field models the intensity of every ray of light flowing through every point in a given scene. Capturing light fields by using adequate equipment is interesting, for it allows significant development and applications. Nevertheless, light fields are very computationally expensive. We address this concern in this thesis through view synthesis. We thus present three contributions. First, we focus on view synthesis given one input image. We are interested, in particular, in studying the stereoscopic case, and by the means of an advantageous combination of neural networks, we present a method that compares favorably with state-of-the-art, all while being lightweight and capable of quick adjustments to new datasets. Then, we extend this contribution to bidimensional contents, by generating light fields this time. We also improve our solution by adding an adversarial component, and we develop a stereo to light fields module allowing to train our approach on stereo data, while generating in the end quality light fields. Finally, we use recurrent neural networks to generate light fields from a free number of input views, by adopting a MPI representation. Our method is lightweight, efficient and can be applied to any depth plane distribution at test time.