3D scene reconstruction from images

Michaël Ramamonjisoa

École des Ponts ParisTech

THÈSE DE DOCTORAT

de l'École des Ponts ParisTech

3D Scene Reconstruction from Images

École doctorale École Nationale des Ponts et Chaussées ParisTech: Mathématiques

Mathématiques Appliquées

Thèse préparée au sein du LIGM-IMAGINE

Thèse soutenue le 22 novembre 2022, par

Michaël Ramamonjisoa

Composition du jury:

Raoul, DE CHARETTE Inria	*Rapporteur, Président du Jury*
Jänne, HEIKKILA Université d'Oulu	*Rapporteur*
Anelia, ANGELOVA Google	*Examinateur*
René Ranftl Epic Games	*Examinateur*
Vincent, LEPETIT École des Ponts ParisTech	*Directeur de thèse*

École des Ponts ParisTech
LIGM-IMAGINE
6, Av Blaise Pascal - Cité Descartes
Champs-sur-Marne
77455 Marne-la-Vallée cedex 2
France

Abstract

Images captured by cameras have become ubiquitous. Being able to reconstruct 3D scenes only using these images would be a highly desirable capability. However, this is very challenging, as images are 2D snapshots of the world, therefore generating ambiguities when lifting them from 2D back to 3D. In this thesis, we focus on two methods for 3D scene reconstruction: single-image depth estimation, and primitive decomposition.

Single-image depth estimation (SIDE) refers to the ability to reconstruct the visible 3D surface of a scene given only a single image as input. This is an ill-posed problem, since many 3D scenes can explain the observed image. In order to solve this problem, modern works rely on data-driven methods and are mainly deep-learning based. These methods therefore use large training datasets of RGB-D pairs, i.e. aligned color and depth images, along with deep neural networks, in order to learn a good prior to predict depth from a single image.

Primitive decompositions can be used to represent a scene as an arrangement of elementary shapes. In the 1960s, Lawrence G. Roberts proposed to represent the 3D world as an arrangement of cuboids. This representation is particularly useful for its simplicity and compactness, which downstream applications such as robotics could leverage.

The first contribution of this thesis introduces a solution to a notorious problem in single-image depth estimation; most methods suffer from smooth edges around occlusion boundaries, while they are supposed to be sharp. Our method, called SharpNet, introduces geometric constraints with synthetic data during training in order to predict sharp depth maps, guided by occlusion boundaries and surface normals.

Our second contribution extends our pursuit of sharper depth edges in SIDE. Neural networks are notoriously biased towards low frequencies, implying that sharp edges, which correspond to high frequency details, will often be overlooked by deep-learning based methods. In this work we introduce a new depth refinement method that sharpens predicted depth maps, and that is able to estimate crisp high frequency details. This method predicts displacement fields, which are used to sharpen depth edges by moving pixels in 2D space. When put on top of baseline single-image depth estimation methods, our method consistently improves the sharpness of depth maps without sacrificing their accuracy.

Our third contribution also aims to improve existing SIDE methods with a simple extension. Most of these previous work rely on U-shaped Encoder-Decoder architectures, often referred to as UNets. While our two first contributions focus on accuracy along occlusion boundaries, this third contribution focuses on efficiency. These occlusion boundaries are indeed usually sparse in natural scenes, which creates an imbalance

that results in smooth predicted depth edges. In WaveletMonoDepth, we instead take advantage of this sparsity. Because depth edges are sparse, we can compute convolutions only in places with large depth variations i.e. occlusion boundaries. Using wavelet decomposition as an intermediate representation for depth maps, we obtain large gains in efficiency while suffering only a minimal loss in accuracy.

Our final contribution explores primitive decompositions as a representation for 3D scenes. RGB-D cameras can be used to scan scenes and store them as 3D point clouds. However, this process is often noisy, and 3D point clouds are expensive to store. With MonteBoxFinder, we propose to represent noisy 3D point clouds with 3D cuboids, by first detecting many 3D cuboids candidates, then finding an arrangement that best fits the scene. This search problem is highly combinatorial, and an exhaustive search is often prohibitive. MonteBoxFinder therefore draws inspiration from Monte Carlo Tree Search methods, in order to efficiently find a good set of cuboids. These cuboids can then be used as ground truth to train single-image cuboid decomposition method.

Résumé

Les images sont aujourd'hui omniprésentes. Pouvoir reconstruire des scènes en 3D à partir d'images est un important défi. Cependant, les images sont une projection en 2D du monde, rendant ambigüe la projection inverse vers la 3D. Dans cette thèse, nous nous intéressons principalement à deux méthodes de reconstruction 3D: l'estimation de profondeur à partir d'une seule image, et la décomposition en primitives.

L'estimation monoculaire de profondeur (SIDE) définit la capacité à reconstruire la surface 3D visible d'une scène étant donnée une seule image entrée. Ce problème est ambigü, étant donné que plusieurs scènes 3D peuvent produire la même image. Les travaux récents utilisent reposent alors sur des méthodes orientées-données, et principalement des méthodes de d'apprentissage profond, utilisant de grands jeux de données composés de paires d'images couleur et de carte de profondeur, ainsi que des réseaux de neurones profonds afin d'apprendre un bon à-priori pour la prédiction monoculaire.

Les décompositions en primitives peuvent être utilisées pour représenter une scène comme un arrangement de formes élementaires. Dans les années 60, Lawrence G. Roberts propose de décrire le monde comme un arrangement de cuboïdes. Cette représentation est simple et compacte, ce qui peut s'avérer utile pour des applications comme la robotique.

En première contribution de cette thèse, nous proposons une solution à un problème récurrent en SIDE: la plupart des méthodes produisent des contours d'occultation flous, alors que ceux-cis devraient être nets. Notre méthode SharpNet introduit des contraintes géométriques ainsi que l'utilisation de données synthétiques pour prédire des cartes de profondeur plus nettes.

Notre deuxième contribution poursuit notre quête de netteté pour les contours d'occultation. Les réseaux de neurones sont connus pour être biaisés vers les basses fréquences, ce qui explique que les contours, étant des détails à haute fréquence, sont souvent ignorés par les méthodes utilisant l'apprentissage profond. Notre nouvelle méthode de correction de carte de profondeur permet d'estimer des cartes plus nettes. Nous prédisons des champs de déplacements afin de déplacer les pixels dans ces cartes. Cette méthode génère des contours nets, sans sacrifier la précision des méthodes qu'elle corrige.

Notre troisième contribution est aussi une simple extension améliorative des méthodes de SIDE. La plupart de ces méthodes utilisent une architecture de type UNet. Alors que les deux premières contributions améliorent la précision des méthodes de SIDE autour des contours d'occultation, cette troisième améliore leur efficacité. Ces contours étant généralement parcimonieux, cela génère un déséquilibre qui résulte le plus souvent en des contours de profondeur flous. Avec WaveletMonoDepth, nous utilisons cette parcimonie

à notre avantage: nous pouvons alors calculer les convolutions uniquement dans les zones à forte variation de profondeur, principalement autour des contours d'occultation. La décomposition en ondelettes est utilisée comme représentation intermédiaire, et permet de générer de forts gains en efficacité, au prix d'une faible perte en précision.

Notre dernière contribution explore la décomposition de scène en primitives 3D. Les caméras RGB-D peuvent être utilisées pour scanner des scènes et les stocker sous forme de nuages de points 3D. Cependant, ce procédé est souvent bruité, et couteux en stockage. Grâce à MonteBoxFinder, nous représentons ces nuages des points bruités sous forme d'arrangement de cuboïdes. Nous détectons d'abord un large nombre de cuboïdes, avant d'en extraire un arrangement qui représente convenablement la scène. Nous nous inspirons de l'algorithme Monte Carlo Tree Search pour résoudre ce problème combinatoire et obtenir de bonnes décompositions en cuboïdes. Les cuboïdes ainsi obtenus peuvent alors servir d'annotations pour entrainer des algorithmes de décomposition en cuboïdes à partir d'une image.

Remerciements

Ces quatre années de thèse ont été chargées en défis et rebondissements, jalonnées de sacrifices, mais aussi ponctuées par des moments de joie. Cette aventure, je ne l'aurais jamais commencée, ni terminée sans la rencontre, le soutien et la confiance des personnes formidables qui m'ont entourée depuis mes débuts dans le monde fantastique de l'informatique et de l'intelligence artificielle.

Vincent, merci beaucoup pour ta confiance dès mes débuts dans la recherche, depuis mon stage MVA jusqu'à la fin de ma thèse. Ta bienveillance, ta recherche du meilleur pour tes étudiants et ton soutien face aux défis qui ont ponctué ma thèse, tels que le COVID ainsi que mes stages, mais surtout notre transfert depuis Bordeaux aux Ponts, font de toi un excellent encadrant et mentor. Notre arrivée à l'ENPC a été la plus grande opportunité pour ma thèse et pour le reste de ma carrière professionnelle, et je suis infiniment reconnaissant d'avoir pu te suivre dans cette aventure.

Merci à l'Ecole Doctoral MSTIC d'avoir accepté et encadré ma thèse. Merci aussi à mon jury de thèse, Janne Heikkilä, Raoul de Charette, Anelia Angelova et René Ranftl, pour leurs retours et commentaires instructifs. Je souhaite tout particulièrement remercier Raoul de Charette pour avoir présidé mon jury et surtout pour sa disponibilité de dernière minute. Je tiens aussi à remercier l'ENPC et surtout Isabelle pour avoir facilité mon transfert ainsi que mes stages chez Niantic et Facebook.

Cette thèse fût l'occasion de découvrir le monde de la recherche en dehors du cadre académique, et surtout en dehors des frontières françaises. D'abord Londres (enfin presque) : je souhaite remercier Niantic Londres et particulièrement Daniyar Turmukhambetov, Michael Firman et Jamie Watson. *Daniyar, thanks for the trust you put in me and our project, and for your support despite the challenges that COVID and remote work put upon us.* Vient Zurich : un grand merci à l'équipe de Meta Reality Labs Zurich pour leur accueil, encore une fois malgré le travail à distance. *Frederik Warburg, Manuel Lópes Antequera, Anton Obukhov and Yubin Kuang, I really enjoyed our collaboration. I know we'll see each other again (and for some, we already did a couple times!).* Enfin, je suis heureux d'avoir rencontré toutes ces personnes en conférence ou en école d'été, stage ou tout à la fois, en Suisse, aux Etats -Unis, en Corée du Sud, en Sicile ou en Israël: Ivan Shugurov, Sergey Zakharov, Mai Bui, Roman Kaskman, Tolga Birdal, Paul-Edouard Sarlin, Damien Robert, Zoe Landgraf, Katrin Renz, *fun times!*

Mes collègues et amis de labo, tant à Bordeaux qu'aux Ponts, merci pour votre aide, pour ces moments partagés à travers la joie et les deadlines, à travers les conférences et les TDs. Pour Bordeaux, merci à Hugo, Pierre, Pierre-Etienne pour leur accueil. Giorgia, un grand merci à toi tout particulièrement, que ce soit à Bordeaux, Québec,

Séoul, Paris, Venise ou Munich. Pour les Ponts, merci aux vieux : Abdou, Pierre-Alain, Tom, Thibault, Theo, François, Xi, Xuchong, Othman, Philippe, Yang, Liza, Rahima, Marie-Morgane, Simon, Robin. Merci aussi aux plus jeunes: Nguyen, Victor, Georgy, Mathis, Monika, Hannah, Nicolas D., Romain, Lucas, Elliot, Antoine, Yannis, Sonat. Merci aux postdocs Nicolas G. et Nermin. Enfin merci aux permanents : Gül, Mathieu, David, Renaud, Pascal, Bertrand, Chaohui.

Aux prophètes de Chronocam / Prophesee, Chiara, Alex, Artiom, Pierre, Charlou, Florian et Adrien, merci pour nos soirées et escapades de folie. Les Supop', merci d'avoir été là aussi et désolé pour tous mes refus pour se voir à cause du MVA ou des deadlines, même en 11A, je suis heureux de toujours vous compter parmi mes amis. Mes amis depuis toujours ou presque, Marion, Hugo, Anna, Joris, Jean-Luc, merci d'avoir été à mes côtés, et pour votre soutien dans ma reprise d'études et au-delà.

A ma famille, Maman, Papa, Mel, Mamie et Papi, merci pour tout.

Enfin, rien de tout cela, depuis ma reprise d'études jusqu'à la fin de ma thèse, n'aurai eu lieu si je n'avais pas eu l'impulsion, le soutien, et la patience d'une personne : Juline. Merci pour tout ce que tu as fait pour moi et pour ton soutien indéfectible. Cette thèse, je te la dois et te la dédie.

Table of contents

Chapter 1

Introduction

1.1 Goals

The goal of this thesis is to develop methods that improve the performance of algorithms that perform 3D reconstruction from images. In particular, we will focus on two 3D computer vision tasks: (i) monocular depth estimation and (ii) primitive-based decomposition of complex 3D scenes.

Estimating sharp monocular depth maps is a very challenging task. As we will see later, monocular depth estimation (MDE), i.e. the task of estimating the visible 3D structure of a scene given a single color image, is already an ill-posed problem unless tackled under strongly constrained scenarios. Most MDE methods employ deep neural networks along with large datasets to learn a prior that allows them to infer reasonable depth maps. However as we show in Chapter 3 learning a good monocular depth estimator is very difficult due to a strong imbalance between regions with strong discontinuities and smooth regions, resulting is smooth depth edges. We therefore propose a two-step method to solve the aforementioned problem and enforce sharp depth predictions around occlusion boundaries. Secondly, we observe that most methods for monocular depth estimation employ deep convolutional networks, which have limited capacity. This results in a limited bandwidth in output depth maps. This frequency bias or "spectral bias" in fully connected / multi-layer perceptron (MLP) (Rosenblatt (1958)) and convolutional neural networks (CNN) (Fukushima (1988)) has been studied by various previous work (Basri et al. (2020); Geifman et al. (2022); Rahaman et al. (2019); Tancik et al. (2020)), and in our case results in smoother depth edges than the ones captured by sensors. In Chapter 4 we propose a method that circumvents this problem by enabling neural networks to

predict 2D displacements, which can in turn transform smooth input depth maps into sharp ones after applying the displacements.

Improving efficiency of convolutional neural networks for monocular depth estimation can be useful for low power applications, for example estimating depth on portable devices. Typical monocular depth estimation methods employ U-Net by Ronneberger et al. (2015) neural network architectures, a class of CNNs. Although these have been long-standing go-to architectures, they are also inefficient, as they treat every pixel in the image the same way, i.e. computing convolutions at every location in the image. U-Nets however encode information at multiple scales. We therefore propose in Chapter 5 a method that explicitely combines the multi-scale aspect of U-Net with the efficiency of wavelet decomposition, resulting in an efficient architecture that exploits the sparsity of information in depth maps.

Primitive-based decomposition of complex 3D point clouds is a challenging task, but can prove beneficial for a wide range of applications such as robotics, compression, and real or virtual interaction with an environment represented by a 3D point cloud of a scene. The goal of this task is to explain a 3D point cloud, which was reconstructed using a sequence of images from the ScanNet dataset (Dai et al. (2017a)), as a set of primitives that best fit the 3D points. This is particularly challenging in cases where the point cloud is very noisy. In Chapter 6, we propose a two-step method that first detects many proposal primitives, then efficiently filters them such that the output set of primitives satisfies a simple set of constraints. The filtering is done using an efficient MCTS-inspired algorithm, that enables tackling this high-complexity problem.

1.2 Motivations

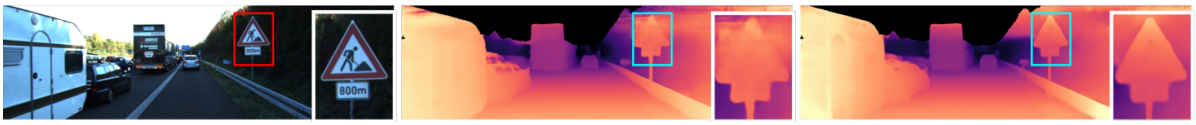
Augmented Reality (AR) aims to blend computer-generated virtual content into a real environment. Different technologies have been used to perform AR. Consumer products perform AR by adding content on frames acquired by cameras, such that its real-time video feed is enhanced with virtual content. Snap Inc's Snapchat filters allows its users to add content to themselves using their phone's front-facing (selfie) camera. Niantic's Pokémon Go (see Figure 1.1) uses smart-phones' back-facing camera to render 3D-aware content to the camera's video feed in real-time. Meta's Quest 2 uses a similar system to achieve AR, using rear-facing cameras mounted on the headset. With the advances in holographic displays development (Xiong et al. (2021)), multiple

Fig. 1.1 3D-aware object insertion in Pokémon Go. In this demo, Niantic Inc. applied Monodepth2 (Godard et al. (2019)) to enable occlusion aware 3D object insertion. https://www.youtube.com/watch?v=7ZrmPTPgY3I&ab_channel=Niantic
.

see-through AR devices have been built. These devices use holographic displays to project 3D content in front of the eyes, while letting light from real world pass through as regular glasses. Microsoft's Hololens therefore enables operators to receive real-time 3D-aware instructions while interacting with the real world, with applications such as construction, manufacturing, surgery or education. While both headsets Quest 2 and Hololens have built-in sensors that can provide 3D information, applications such as Pokemon Go or Snapchat filter require 3D estimation from color images, as most smart-phones are not equipped with 3D sensors. More critically, both stereo-based and time-of-flight sensors provide incomplete or noisy information around objects boundaries. Being able to estimate sharp depth maps around object can therefore prove beneficial for more realistic 3D-aware object insertion.

Virtual Reality (VR). Contrary to AR, VR puts its user in a fully immersive environment, where the only visual information visible to the user is rendered with computer graphics. While many VR applications are developed for static user experiences, i.e. when the user can only move their head, experiences with user in motion require performing 3D reconstruction to avoid collision with obstacles. While headsets usually have embedded depth sensors, these are often noisy and can sometimes fail when reconstructing texture-less or specular surfaces. Being able to estimate depth from color images generates another depth sensor for "free" for VR headsets.

Robotics. Performing 3D scene understanding from images is useful for several robotics applications. For robot navigation, being able to estimate the distance to objects is critical for obstacle avoidance and path planning. For automated manufacturing, being able to estimate 3D in real time is also useful for tasks such as grasping.



RGB BTS [26] Ours

Fig. 1.2 Monocular depth estimation for autonomous driving. Brightest regions are closer to the camera, while darkest ones are further. Figure from AdaBins by Bhat et al. (2021).

Autonomous Driving. When driving a car, a human must always estimate its distance to other road users such as other vehicles and pedestrians, but also its distance to potential obstacles such as trees, traffic lights, etc. This estimation is critical, both for navigation and avoiding accidents. While sonar sensors are widely used in cars to evaluate distance to objects, they are often limited to a few meters range. RADAR and LiDAR sensors are used in Advanced Driver Assistance Systems (ADAS) however they are relatively expensive or bulky compared to color cameras. Estimating reliable depth from images such as the example in Figure 1.2 is therefore a low-cost yet key component to enable safer driving, and even more so for fully autonomous driving.

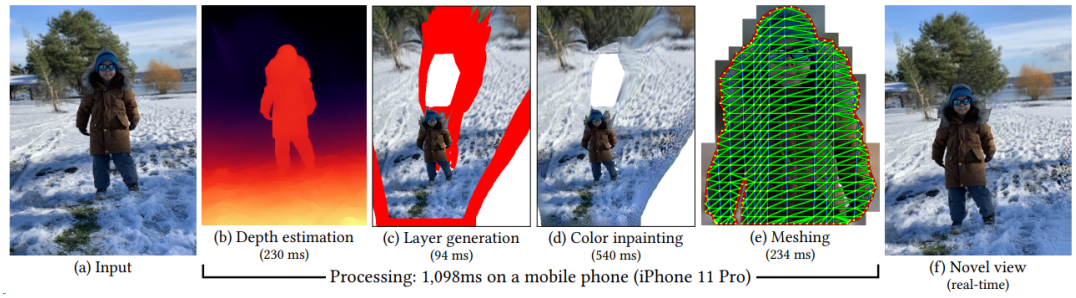
3D photography and image editing are two applications that can benefit from monocular depth estimation. 3D photography aims to generate appealing 3D effects such as motion parallax which changes the 3D viewpoint of the camera, or virtual dolly zoom which changes its focal length (see Figure 1.3). These effects require estimating the 3D scene given only a single photograph. It also requires inpainting regions in the image that are disoccluded by the generated 3D motion. Kopf et al. (2020) perform one-shot 3D photography by leveraging depth estimation and color inpainting to generate novel views of the input image. Similar to augmented reality, 3D photography requires estimating sharp occlusion boundaries with depth estimation for realistic effects.

1.3 Approach and context

Most of earlier works focused on depth perception from stereo pairs (Scharstein and Szeliski (2002)), as triangulation techniques can yield unambiguous 3D reconstructions from matching pairs of points from both images. Another line of works, known as Structure-from-Motion (SfM) (Schönberger and Frahm (2016); Seitz et al. (2006)) focuses on exploiting correspondences in frames taken from video sequences or even multiple cameras to perform 3D reconstruction. However both stereo vision and SfM require multiple images as input to infer 3D geometry. While humans can leverage monocular

(a) Motion parallax results from Shih et al. (2020). More here.

(b) Dolly zoom effect results from Shih et al. (2020). More here.



(c) One-shot 3D photography (Kopf et al. (2020)) pipeline

Fig. 1.3 Application of depth estimation to 3D photography effects. Animations best viewed with Acrobat Reader

visual cues to infer depth even with a single eye (Howard (2012)), performing monocular depth estimation with computers has been a long-standing problem, and performing depth estimation from a single *still* image obtained from a single camera is even more challenging. This task, called Single Image Depth Estimation (SIDE) aims to predict depth at all pixels in a single input image. It is an ill-posed problem because multiple 3D scenes can be projected to the same 2D image, implying many ambiguities as discussed in Section 1.4.

***Single Image* vs *Monocular* Depth Estimation.** In this section and the rest of the manuscript, we interchangeably use Monocular Depth Estimation (MDE) and Single Image Depth Estimation (SIDE) to refer to inference of depth given a *single image* as input. However, MDE is more general, since *monocular* means *single camera*, which means multiple frames of the same camera could potentially be used for inference; this is for example a reasonable assumption in methods performing *Shape-from-X*, such as Shape from Shading (Horn and Brooks (1989); Horn (1975)) or Shape from Defocus (Favaro and Soatto (2005); Subbarao and Surya (1994)). However, in modern litterature, MDE now mainly refers to scenarii with a single frame for inference. Therefore, unless specified otherwise, we use MDE to refer to single-frame-single-camera depth estimation.

Our approach to MDE. Deep-learning based methods have enabled breakthroughs in MDE such as the seminal works of Eigen et al. (2014), Laina et al. (2016) and Zhou et al. (2017). We therefore chose to also leverage deep-learning based methods to perform MDE and aimed to improve the sharpness of their predicted depth edges, as most state-of-the-art methods were then predicting globally accurate depth maps but often with smooth depth edges. This progress was made possible thanks to:

- large scale datasets such as the NYUv2-Depth dataset by Silberman et al. (2012) and PBRS by Zhang et al. (2017),
- efficient hardware such as high-end Graphics Processing Units (GPU),
- widely available open-source research code, such as the popular deep-learning library Pytorch Paszke et al. (2019).

Generating ground truth data for single-view 3D reconstruction by primitive decomposition. Finally, we explore another primitives decomposition as another representation to perform 3D reconstruction from images. In order to generate ground truth cuboids to supervise single-image primitive decomposition algorithms for 3D

reconstruction, we first leverage the RGB-D sequences of ScanNet (Dai et al. (2017a)), which are used to reconstruct a 3D mesh using BundleFusion(Dai et al. (2017b))[1]. The next task is then to fit this mesh with 3D primitives, which we chose to limit to cuboids. This task proved to be very challenging due to the high noise in ScanNet, and we therefore designed a method discussed in Chapter 6 to extract 3D cuboids from the noisy 3D reconstructions. In future work, the generated data could be used to supervise deep learning methods for single-image 3D reconstruction using cuboids.

1.4 Challenges

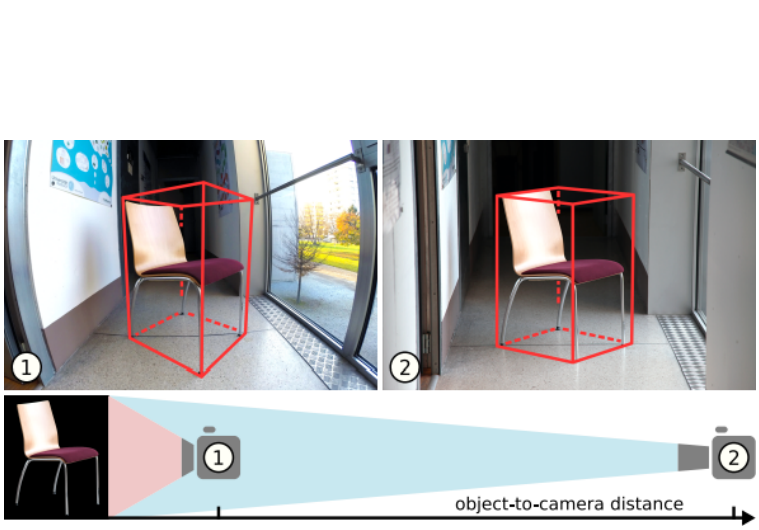
In this section, we discuss the main challenges that arise in 3D reconstruction from images. We first discuss ambiguities that arise in single-image depth estimation (SIDE), before discussing different types of noise in data that we encountered.

1.4.1 Ambiguities in 3D reconstruction from a single color image

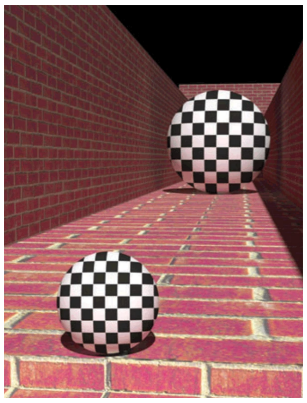
Scale ambiguity can arise in two cases, shown in Figure 1.4: (i) doubling the focal length and distance results in similar appearance, as shown in Figure 1.4a, and (ii) doubling the object's size and distance to the camera also results in similar appearance. In the context of reconstructing a 3D scene given a single image as input, such ambiguities cannot be solved at test time. Learning-based approaches however allow monocular depth estimation methods to learn plausible 3D shapes from training data where the camera intrinsics are known and fixed, such that an object would rarely appear at an abnormal scale. They can also leverage context, such that depth maps are inferred not only from each single object separately, but rather such that the scene is globally consistent. Finally, if multiple images can be used, as is the case for building the ScanNet dataset (Dai et al. (2017a)), the scale ambiguity can be solved with multi-view consistency and the variation in camera intrinsics can be taken into account.

Contextual ambiguity is also a challenge for predicting depth. In the iBims evaluation benchmark, Koch et al. (2020) build adversarial situations, where for example in Figure 1.5 a printed picture of a scene is put on a wall. Because usually a neural network is trained to decode perspective images into 3D geometry through visual cues, a picture containing the same cues will generate the same depth map. Therefore, instead of predicting a flat

[1]This step is performed in the ScanNet original work.



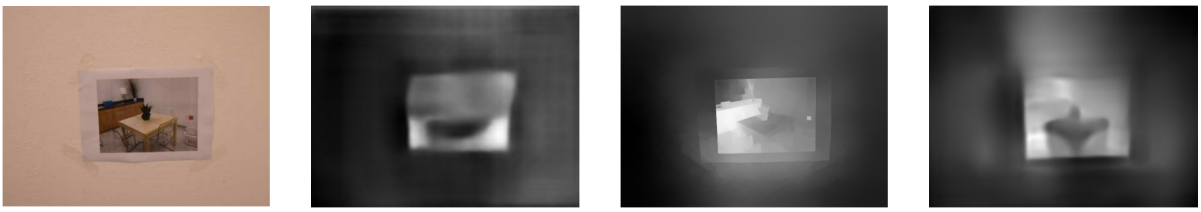
(a) Focal length ambiguity Example from Grabner et al. (2019).



(b) Size ambiguity Example from Sterzer and Rees (2006), scale ambiguity arises from .

Fig. 1.4 Scale ambiguity. In (a) scale ambiguity arises from varying focal length and distance at the same time, while in (b) the object size and distance are changed simultaneously. In both cases the object appears at the same scale in 2D but is captured from very different distances.

surface, the network is fooled and predicts an actual 3D scene. While being an interesting topic, all methods trained using a single image as input still fail to solve such kind of ambiguity, which arises from a lack of context given to the network.

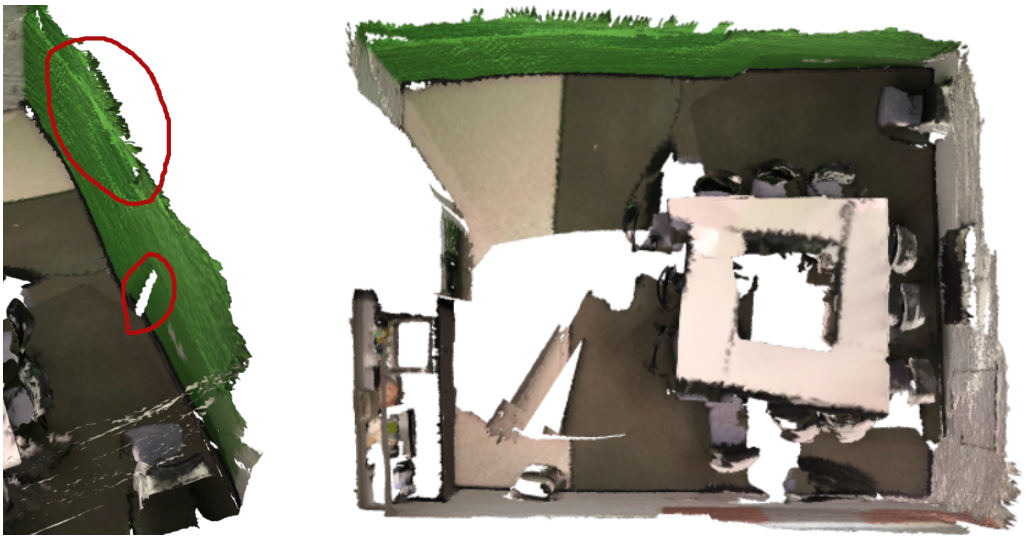


(a) RGB (b) Laina et al. (2016) (c) Liu et al. (2015) (d) Eigen and Fergus (2015)

Fig. 1.5 Example of a visual ambiguity. Deep-learning based methods trained on datasets with RGB-D pairs of indoor scenes decode visual cues into 3D geometry. When a neural network is tasked to predict the depth of (a) a printed image of a scene set on a planar surface such as a wall, it predicts the depth (b-c-d) of the original scene rather than the wall, resulting in a large error. Images from Koch et al. (2018)

1.4.2 Noisy data

Noise is a common challenge in data science. When it comes to 3D reconstruction, noise can come from two main sources: sensors and computation errors. In this section we



(a) Kinect noise **(b)** BundleFusion noise

Fig. 1.6 Inaccuracy in 3D reconstructions. Kinect sensors exhibit noise that creates creases on walls, as shown in Figure 1.6a. When performing 3D reconstruction based on RGB-D sequences and BundleFusion (Dai et al. (2017b)) in ScanNet (Dai et al. (2017a)), reconstruction errors such as for the set of tables in Figure 1.6b can also come from algorithm errors.

cover two types of noise in 3D data that we encountered: inaccurate data and missing data. We summarize these challenges in Figures 1.6 and 1.8.

1.4.2.1 Inaccurate data

Sensor noise. Depth sensors such as Kinect are, as any other real-world sensor, noisy. This noise comes from multiple sources: the structured light emitter, the receiver that reads reflected light, and the matching algorithm which performs disparity -and ultimately depth- measurement.

3D reconstructions performed using noisy RGB-D sequences acquired with Kinect, such as the ScanNet dataset, will then directly inherit the noise from sensors, as shown in Figure 1.6a. Finally, another source of inaccuracies in such 3D reconstructions would be the reconstruction algorithm itself, i.e. BundleFusion (Dai et al. (2017b)) in the case of ScanNet, as shown in Figure 1.6b.

1.4.2.2 Missing data

Structured-light matching. In the case of depth sensing, missing data can also come from multiple factors. For depth acquired using a Kinect sensor, which uses

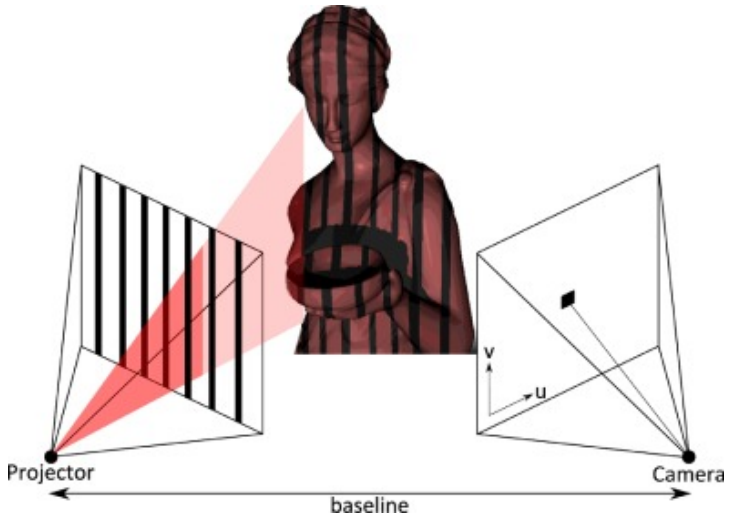


Fig. 1.7 Depth estimation using structured light is performed by triangulation between a sensor (camera) and a projector. A known pattern is projected onto a 3D scene, and the sensor observes a deformed pattern. By performing matching, the 3D surface can be reconstructed via triangulation (the baseline distance B between the projector and the camera is known and calibrated). However, when the object occludes the pattern, matching cannot be performed, resulting in missing depth values. Figure from Sarbolandi et al. (2015).

matching-based disparity measurement with structured light (see Figure 1.7), missing data occurs in two major cases: (i) shadows by occlusion, i.e. when part of the projected structured light used for matching becomes occluded by the scene and cannot therefore be recovered, and (ii) when structured light is projected on a specular (i.e. reflective) surface such as the marble table. In the latter case, no light is reflected back towards the receiver sensor, making it impossible to perform matching. We show an example of such missing data in Figure 1.8a.

3D reconstructions that are performed by scanning scenes with RGB-D sensors, as done in ScanNet, also inherit from these issues. However the most common source of missing data in this case is human: some parts of a 3D scene are either not scanned because of inaccessibility, or by mistake. An example of such incomplete data is shown in Figure 1.8b. In the case depth estimation, there are two main causes of noise: (i) the sensor noise, which varies depending on the sensor but translates in noisy measurements of depth for each pixel, (ii) missing data due to disocclusions, i.e. the situation where part of a scene is hidden in one view of the multi-view sensor . 3D point clouds inherit directly from the noise of the acquisition system. Such noise can come from sensor imperfection (e.g. in the case of time-of-flight sensors, noise on time measurements), or occlusions.

(a) NYUv2 **(b)** ScanNet

Fig. 1.8 Incompletion in 3D reconstruction with RGB-D data. (Left.) In NYUv2 (Silberman et al. (2012)), missing depth data (bottom, in black) comes from the Kinect sensors relying on structured-light and matching, which is impossible to perform if the projected pattern is occluded by objects or if the pattern is projected on a specular surface. **(Right.)** In ScanNet (Dai et al. (2017a)), missing data can also come from human error, i.e. the human not fully scanning the 3D scene.

1.5 Outline

The outline of this thesis is organized as follows:

Chapter 2 Related work We first present an overview of related previous work in 3D reconstruction from images, with a particular focus on learning-based single image depth estimation. We also cover popular representations for 3D data that were used in several of the aforementioned related works, along datasets that are commonly used for 3D reconstruction, before covering single-image depth estimation related works in detail.

Chapter 3 Predicting Sharp Occlusion Boundaries in Monocular Depth Estimation with Geometry Guidance This chapter introduces the first contribution of this thesis, which set a new state-of-the-art on sharpness in monocular depth estimation, using synthetic data and geometric constraints.

Chapter 4 Estimating Sharper Depth Maps with Displacement Fields This chapter introduces a new class of network for monocular depth estimation, which is able

to circumvent the low-frequency bias of neural networks using displacement fields to sharpen depth estimates.

Chapter 5 Improving the Efficiency of Monocular Depth Estimation using Wavelets Transforms This chapter introduces a method that exploits the structure of depth maps to improve the efficiency of U-Net (Ronneberger et al. (2015)) architectures for monocular depth estimation. We show that pairing wavelet decompositions with this architecture yields improves efficiency of state-of-the-art methods without sacrificing accuracy.

Chapter 6 Reconstructing 3D Scenes as Complex Sets of Primitives In this chapter, we propose a method that first extracts a large set of cuboids from a noisy 3D point cloud that was obtained a sequence of RGB-D images, then efficiently filters them to return a set of cuboid that best fits the point cloud.

Chapter 7 Conclusion In this final chapter, we reflect on our contributions and summarize this thesis takeaways. We then propose some future research perspectives which could build upon our contributions.

1.6 List of Publications

In this thesis, we cover four papers:

- **Michaël Ramamonjisoa**, and Vincent Lepetit, (2019) SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation, In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*,
- **Michaël Ramamonjisoa**, Yuming Du, and Vincent Lepetit, (2020) Predicting Sharp and Accurate Occlusion Boundaries in Monocular Depth Estimation Using Displacement Fields, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,
- **Michaël Ramamonjisoa**, Michael Firman, Jamie Watson, Vincent Lepetit, Daniyar Turmukhambetov (2021) Single Image Depth Prediction with Wavelet Decomposition, In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,

- **Michaël Ramamonjisoa**, Sinisa Stekovic, Vincent Lepetit, (2022) MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud, In *European Conference on Computer Vision (ECCV)*.

We made the code for all projects open-source[2345]. We also created explanatory videos for **DisplacementFields**[6] and **WaveletMonodepth** on their respective webpages. Our research also led to the release of open-source datasets. We released the NYUv2-OC dataset alongside the publication of SharpNet. Later upon publication of DisplacementFields, we released NYUv2-OC++ as an extension to NYUv2-OC. Finally, in the context of a collaborative project with Dr. Giorgia Pitteri, we released the Synthe-TLess dataset[7].

I was fortunate to be given the opportunity to present my work in various research groups such as LaBRI (Université de Bordeaux), Niantic, Meta Reality Labs, Microsoft Cognition and Google Zurich. I also presented these works during poster sessions at ICCV 2019, CVPR 2020, CVPR 2021, and at the International Computer Vision Summer School (ICVSS) 2022.

During my PhD, I also participated to the following collaborative works, which are not discussed in this manuscript:

- Giorgia Pitteri, **Michaël Ramamonjisoa**, and Vincent Lepetit, (2019) On Object Symmetries and 6D Pose Estimation from Images, In *International Conference on 3D Vision (3DV)*,
- Van Nguyen Nguyen, Yuming Du, Yang Xiao, **Michaël Ramamonjisoa**, and Vincent Lepetit, (2022) PIZZA: A Powerful Image-only Zero-Shot Zero-CAD Approach to 6DoF Tracking, In *International Conference on 3D Vision (3DV)*,
- Frederik Warburg, **Michaël Ramamonjisoa**, and Manuel López Antequera, (2022) SparseFormer: Attention-based Depth Completion Network, In *Proceedings of the*

[2] https://github.com/MichaelRamamonjisoa/SharpNet
[3] https://github.com/dulucas/Displacement_Field
[4] https://github.com/nianticlabs/wavelet-monodepth
[5] https://github.com/MichaelRamamonjisoa/MonteBoxFinder
[6] https://michaelramamonjisoa.github.io/projects/DisplacementFields
[7] https://github.com/MichaelRamamonjisoa/SyntheT-Less

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

Chapter 2

Literature review

In this chapter, we review existing work related to the two tasks studied in this thesis: (i) monocular depth estimation, and (ii) fitting 3D point cloud with primitives. In Section 2.1, we first start with a brief overview of deep learning and its main components, as methods described in Chapters 3-4-5 are deep learning based. In Section 2.2, we then describe popular representations 3D scenes, among which are depth maps which we use extensively, and primitives decompositions which we use in Chapter 6. Finally, in Section 2.3 we review previous work on learning-based single image depth estimation.

2.1 Deep Learning

The first three contributions of this thesis, presented in Chapter 3, Chapter 4 and Chapter 5 rely on deep learning to perform various tasks related to monocular depth estimation. We therefore review important aspects of deep learning in the following section. We first cover a brief history of deep learning in computer vision, then discuss its key components, and finally discuss in more detail the U-Net (Ronneberger et al. (2015)) neural network architecture, which we use throughout the thesis.

2.1.1 The rise of Deep Learning in computer vision

As many other fields of The vast majority of computer vision tasks today are tackled using deep-learning-based techniques. For most computer vision tasks, deep-learning-based methods have at least improved performances over classical methods. In a large and growing number of instances, deep-learning-based methods enabled breakthroughs, and are often the only way to solve notoriously hard computer vision problems. Although it had originally been studied in early years of computer vision (Rosenblatt (1958)), the

recent surge of deep learning for computer vision can be attributed to two main factors: (i) the collection and public release of large scale datasets such as ImageNet (Deng et al. (2009)), (ii) the steady increase in development of Graphics Processing Units (GPU) and (iii) the development of breakthrough GPU-trained convolutional neural networks (CNN) (Fukushima (1988)) architectures such as AlexNet (Krizhevsky et al. (2012)) and ResNet (He et al. (2016a)), which set the path to many more breakthroughs across different sub-fields of computer vision.

2.1.2 Components

For a detailed introduction to deep learning and its components, we refer the reader to Goodfellow et al. (2016). In this section, we only review the key components required to build a deep-learning-based method in order to solve a particular task.

Dataset. A large dataset is necessary to train a deep neural network. This dataset is made of a set of training samples $\mathcal{X}$ which in the case of labeled datasets, consist in pairs $\{x_i, y_i\}$ where x_i is the data, typically an image in computer vision applications, and y_i is a label, which can be a single value, or a vector, array, or volume of values, depending on the application. For classification, y_i is a single value that indicates the ground truth class of an image x_i. For dense regression or classification tasks such as depth estimation or semantic segmentation respectively, the label y_i is an array of values corresponding to per pixel ground truth value for depth or class. These labels, also known as annotations, are notoriously hard and expensive to obtain, and often come with noise that usually originates from human error or sensor noise.

Neural network architecture. A neural network architecture $\mathcal{F}_\theta$ is a parametric family of differentiable functions. This differentiability is essential in order to allow the network to update its parameters θ, also known as weights, over training based on an measured loss function and a chosen optimization method through *backpropagation* (Rumelhart et al. (1986)). Popular neural network architectures for vision include convolutional neural networks (CNNs) (Fukushima (1988)), multi-layer-perceptrons (MLP) by (Rosenblatt (1958)), and more recently transformers (Vaswani et al. (2017)) which have been introduced to vision with works such as Vision Transformer (ViT) (Dosovitskiy et al. (2020)) or DETR (Carion et al. (2020)).

Loss function. In order to train a neural network, it is necessary to define a function ℓ that evaluates the deviation between the neural network's outputs and its target

outputs. In the case of supervised learning, i.e. when ground truth annotations y_i are provided, the target outputs are the annotations themselves. As we will see later, it is also possible to define loss functions for self-supervised learning, i.e. when no ground truth annotations y_i are provided, such that by optimizing this loss function, the network learns to produce better outputs for the task at hand. In all cases, the loss function should also be differentiable, such gradient descent methods can be applied to optimize the neural network's parameters.

Optimization method. Optimizing a loss function $\mathcal{F}$ across all samples of a training set is generally intractable, both due to computational limits and in general non-convexity of the loss function. Instead, most deep-learning-based method rely on Stochastic Gradient Descent (SGD) optimization techniques. SGD methods for deep learning usually follow four steps:

- a *mini-batch* $\mathcal{X}_B = \{x_i, ..., x_{i+n}\}$, which consists of a subset of the N training samples, is randomly sampled from the training data,
- the network $\mathcal{F}_\theta$ performs predictions on all input data in $\mathcal{X}_B$,
- the loss function ℓ is evaluated to measure the deviation between the outputs $\{\mathcal{F}_\theta(x_i), ..., \mathcal{F}_\theta(x_{i+n})\}$ and the target outputs $\mathcal{Y}_B = \{y_i, ..., y_{i+n}\}$,
- the gradient of the loss function for all samples in the mini-batch w.r.t the networks parameter θ are aggregated in $\nabla_\theta \ell$ – usually by taking the average of the individual gradients – and $\nabla_\theta \ell$ is back-propagated to update the networks weights θ. Since all components in the neural network are differentiable, the chain-rule is applied to obtain the gradient of ℓ w.r.t. θ.

Several variants have been developed to improve the stability and performance of SGD, for example the use of momentum (Nesterov (1983)), RMSProp (Tieleman et al. (2012)), or the most popular Adam optimizer (Kingma and Ba (2015)), which we use extensively in this thesis.

2.1.3 U-Net

Since its introduction by Ronneberger et al. (2015), the U-Net architecture, shown in Figure 2.1 has been the standard basic structure for dense prediction applications in computer vision. In this section briefly explain the task of dense prediction with neural networks, and cover the key components of the U-Net architecture, as well as its variants.

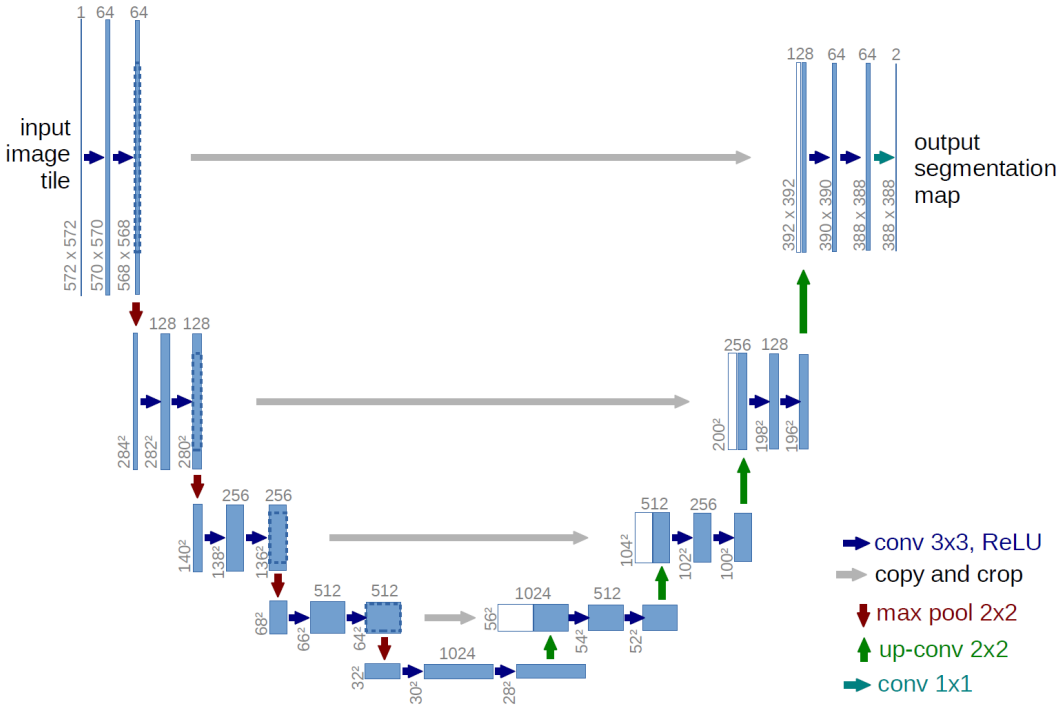


Fig. 2.1 Overview of the U-Net architecture introduced by Ronneberger et al. (2015)

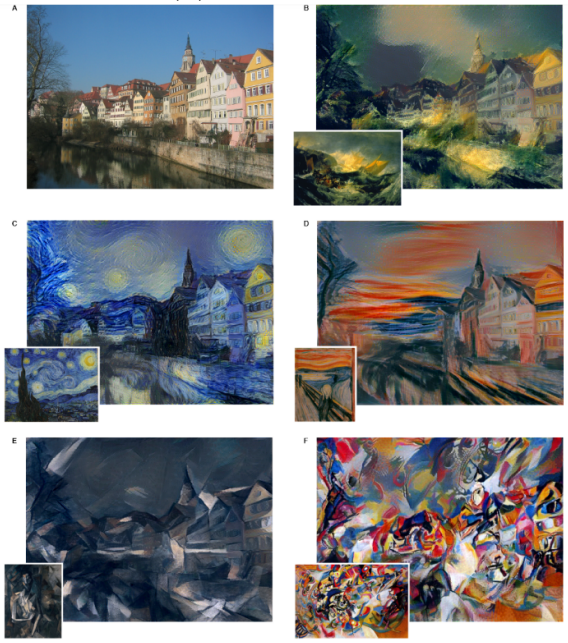
Dense prediction with neural networks. Contrary to classification or detection, dense prediction tasks require the network to predict outputs at the size of the input data. In the most common case of an image as input, the network is also tasked to predict an image at its output, where each pixel of the output image must correspond to a value in the output space. A few example tasks are shown in Figure 2.2.

Encoder-Decoder. The U-Net has an encoder-decoder structure. The *encoder* first extracts a multi-resolution representation through a cascade of convolutions and down-sampling operations. Such class of encoders include the seminal ResNet (He et al. (2016a)), VGG (Simonyan and Zisserman (2015)), or AlexNet (Krizhevsky et al. (2012)) works. The *decoder* then performs decoding and upsampling operations, where the multi-resolution feature maps extracted by the encoder are run through a sequence of convolutions, non-linearities, and upsampling operations.

Skip-connections. Residual connections, also known as *skip-connections* are usually employed in U-Net-based architectures. While in ResNet (He et al. (2016a)), skip connections are short range, the skip connections in U-Net are long range. This ensures

(a) Denoising from RIDNet (Anwar and Barnes (2019))

(b) Semantic segmentation from PSPNet (Zhao et al. (2017))



(c) Style transfer from Gatys et al. (2015)

(d) Depth estimation with MiDaS (Ranftl et al. (2020))

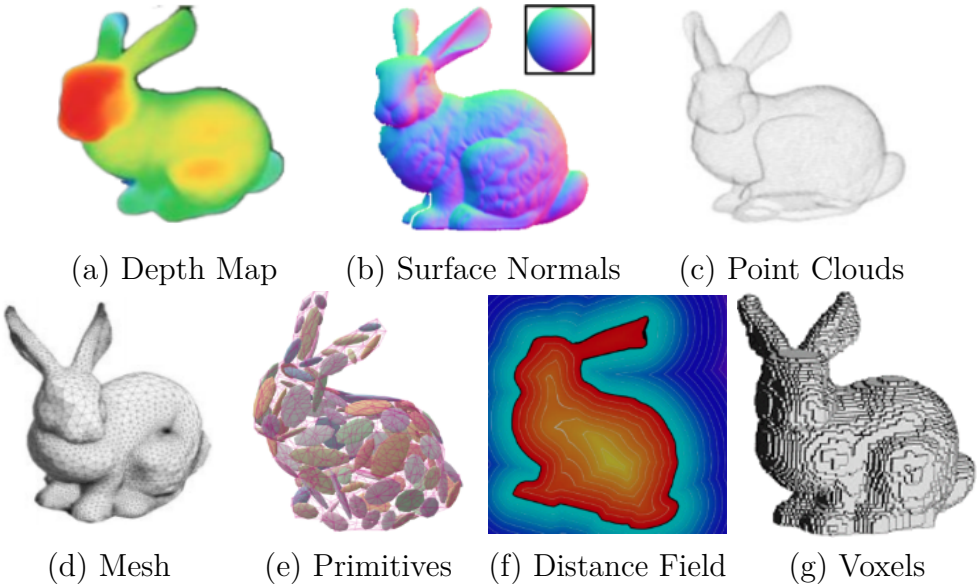
Fig. 2.2 Examples of dense prediction tasks.

high-resolution content is maintained during the decoding process, which starts from the low-resolution bottleneck and progressively upsamples the feature maps.

Alternatives to U-Net. While U-Net are still the primary choice for dense prediction tasks using neural networks, some alternatives have also been developed. For example, two-stage architectures such as Mask-RCNN (He et al. (2017a)) first detect object bounding boxes then segments individual object before merging them into a dense segmentation of the full image. Very recently, transformers architectures have been leveraged by Ranftl et al. (2021) for dense predictions based on ViT (Dosovitskiy et al. (2020)).

2.2 Representing 3D Scenes

In this section we present several ways to represent 3D scenes. We first present explicit representations such as point clouds, depth maps, voxels, and primitive decompositions, which we present an overview in Figure 2.3. We then discuss implicit scene representations formed by neural fields.



(a) Depth Map (b) Surface Normals (c) Point Clouds

(d) Mesh (e) Primitives (f) Distance Field (g) Voxels

Fig. 2.3 Overview of some popular 3D representations. This figure presents the popular 3D representations that we cover in this paper. The depth maps and surface normals are 2.5D representations (single-view surfaces), while point clouds, meshes, primitive decompositions and distance fields are used to represent full 3D shapes. Mots images were taken from Thomas Funkhouser's presentation at the 3DGV 2020 seminar.

2.2.1 Point Clouds

Point clouds are arguably the most straight-forward 3D representation. They encode geometry as a set of points sampled in 3D space. A point cloud $\mathcal{S}$ can be therefore written as $\mathcal{S} = \{X_i \in \mathbb{R}^3\}_{i=1..N}$ where N is the number of points. These points can also have an associated normal, especially when representing 3D surfaces from which they are sampled. Due to their simplicity, point clouds are a popular representation for a large number of works in 3D computer vision, such as 3D semantic segmentation, following the seminal works PointNet and PointNet++ (Qi et al. (2017a,b)) for permutation invariant processing of point clouds with MLPs. Given their success for processing images, several works focused on adapting popular operators used in deep learning architectures to work with point clouds, including convolutions (Boulch (2020); Li et al. (2018); Thomas et al. (2019); Xu et al. (2018b)) and more recently transformers (Guo et al. (2021); Zhao et al. (2021)).

2.2.2 Depth Maps

Depth maps are a common 3D representation, which we use extensively in our works. In this section we first define depth maps, then explain how they can be used to produce 3D point clouds, and finally present common ways to visualize them. We dedicate a more complete section to cover state-of-the-art in depth estimation in Section. 2.3.

Definition. Depth maps are 2D projections of 3D scenes onto a camera's image plane. Assuming a pinhole camera model, with intrinsic matrix $\mathbf{K}$, the 3D projection model is as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = (1/Z)\mathbf{K}\boldsymbol{P} = \mathbf{K} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{P} = (X, Y, Z)^{\mathrm{T}}$ is a 3D point in camera coordinates frame. Z is then the depth value at pixel (x, y) in the resulting depth map. In Figure 2.4, we show a visualization of this projection. Two points P_1 and P_2 reproject in different pixels, the depth map value at these pixels corresponds to their coordinate projection along the camera optical axis.

Back-projection Recovering a 3D point cloud from a depth map is also possible using back-projection. Given a depth map $(x, y) \mapsto Z_{(x,y)}$, which we will write as $\boldsymbol{Z}$ for

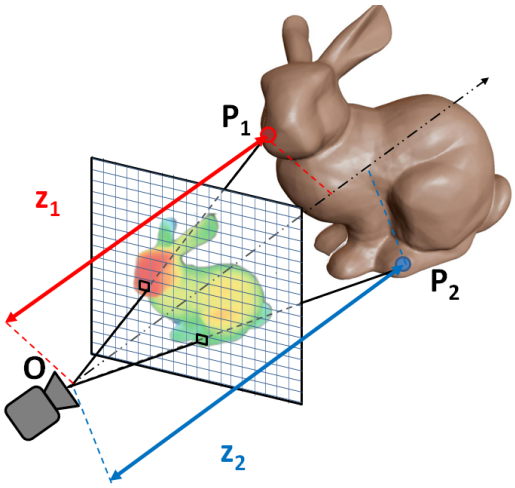
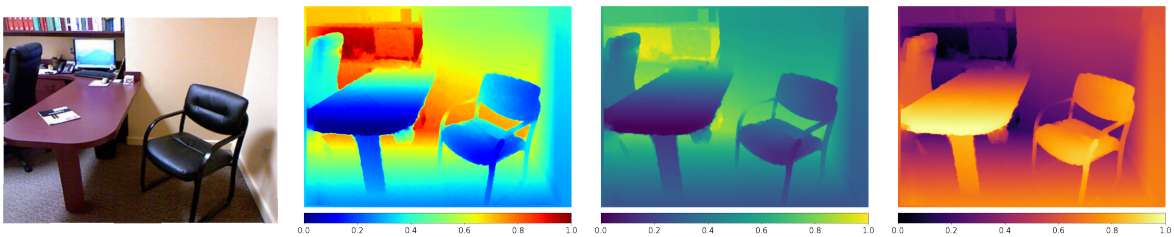


Fig. 2.4 Formation of depth maps. Two 3D points P_1 and P_2 project onto two different pixels, which take as depth values the distance along the camera's optical axis. JET color map is used in this visualization, where closest regions appear more **red**.

simplicity, we can obtain a 3D point $P = (X, Y, Z)$ from a pixel (x, y) and its associated depth value $Z_{(x,y)}$ using:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Z_{(x,y)} \begin{bmatrix} X/Z_{(x,y)} \\ Y/Z_{(x,y)} \\ 1 \end{bmatrix} = Z_{(x,y)} \cdot \mathbf{K}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.2}$$

This operation is useful for example for re-projection of a depth map or its associated color image onto another camera image plane by applying a transform $[R|T] \in \mathrm{SE}(3)$ to P then Equation (2.1), which is for example the case in Godard et al. (2019, 2017).



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

Fig. 2.5 Colormaps for depth maps. Different color maps are used to draw depth maps in the litterature as well as in this manuscript. (b-c-d) are the same depth map obtained using a Kinect sensor, and are paired with the RGB image (a) in the NYUv2-Depth dataset (Silberman et al. (2012)). For the JET (b) and VIRIDIS (c) colormaps, the right-most part of the colormap indicates the farthest regions. For the INFERNO colormap (d), we represent inverse-depth, so brightest regions are the closest to the camera.

Visualization Since depth maps are represented as a 2D arrays of scalar values, they can be visualized as an images. While grayscale 2D images are an option to visualize depth maps, the community has opted to convert these grayscale images to *pseudo-color* images, i.e. mapping one dimensional values to RGB colors. These type of mappings are known as *color mappings*, and are used to improve the perception of information. Different color maps are used for different applications, and depend on the field. A popular example of color map is the mapping of temperature, where hot temperatures are usually mapped to red colors, and cold temperatures are mapped to blue colors. Different color maps have been used for visualizing depth maps, for example JET, VIRIDIS, PLASMA or INFERNO. As shown in Figure 2.5 we use JET, VIRIDIS and INFERNO respectively in Chapter 3, Chapter 4 and Chapter 5.

2.2.3 Surface Normals

Similar to depth maps, surface normals maps encode visible surface geometry. Normal maps are projections of 3D surface normals on an object's surface (see Figure 2.6). Surface normals and depth maps are also strongly related, as the former are the 3D derivative of the latter. This relationship between depth and normals has been used extensively in a number of previous work to perform joint constrained predictions (Qi et al. (2018); Ramamonjisoa and Lepetit (2019); Yang et al. (2018b)). Surface normals are also useful for 2.5D processing, such as image inverse rendering (Li et al. (2020b); Sengupta et al. (2019); Wang et al. (2021); Yu and Smith (2019); Zhu et al. (2022)). Visualizing normal maps is usually done using normal mapping, which maps the 2D angular coordinate (elevation/yaw, azimuth/pitch) of normal vectors to RGB values. In Figure 2.7, the normal mapping convention is shown on the projection of a sphere, along with examples of normal maps.

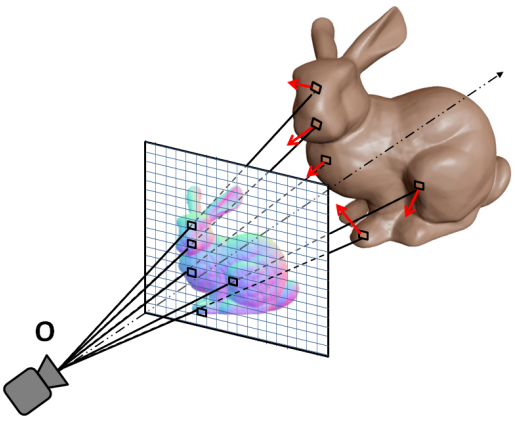


Fig. 2.6 Formation of normal maps. Normal maps are the projection of the values of 3D surface normals (**red arrows**), from the 3D camera coordinate frame, onto the camera image plane.

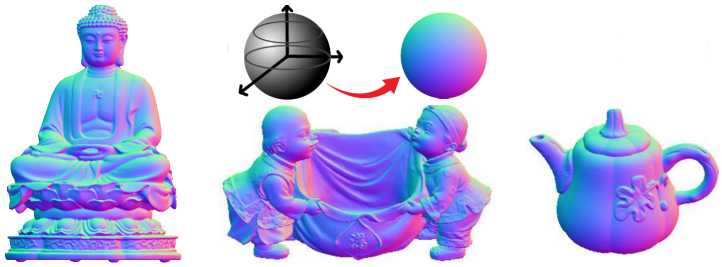


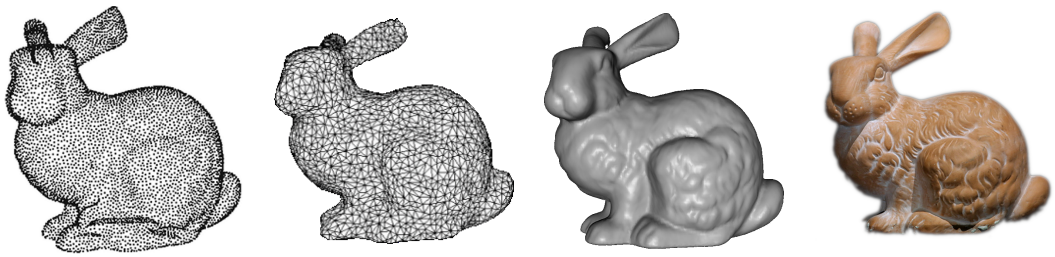
Fig. 2.7 Examples of normal maps. The normal mapping of yaw and pitch angles (spherical coordinates) is shown at the top.

2.2.4 Meshes

Meshes introduce connectivity between points in point clouds. Points are commonly renamed as *vertices* V, and are connected by *edges* E. Edges can be constructed to form a graph (V, E) from which cliques can be used to define *faces* F. The most common 3D meshes usually use triangle or quadrilaterals, i.e. 3- or 4-point cliques respectively, in order to represent 3D surfaces. Normals can also be computed for faces and vertices to produce a more complete 3D surface representation, which is for example needed to produce appealing renderings of a mesh, as shown in Figure 2.8. Compared to point clouds, meshes have a topology that can be exploited for 3D reconstruction using deep learning. Wang et al. (2018b) designed a method that reconstructs an object from an input image by deforming a spherical mesh, using graph convolutions. One downside of mesh deformation to reconstruct a 3D shape is that it does not allow topology changes. Groueix et al. (2018) formulated 3D shape reconstruction as deformations of a set of patches therefore enabling reconstruction of shapes with any topology. Pan et al. (2019) tackled the topology adaptation problem in template deformation by allowing face removal in the template mesh.

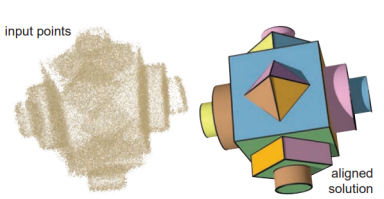
2.2.5 Primitive Decompositions

3D scenes can also be decomposed into simple primitives to enable higher-level understanding of the scene, but also compact representations for efficient computer graphics, 3D CAD model design, or robotics. As shown in Figure 2.9, such primitives can include for example planes, cylinders, spheres, cuboids, or ellipsoids, but these primitives can also be learned (Deprelle et al. (2019)). While planes (Li et al. (2011); Liu et al. (2019, 2018); Monszpart et al. (2015); Schnabel et al. (2007)) and cuboids (Niu et al. (2018); Tulsiani et al. (2017); Zou et al. (2017)) decompositions dominated 3D computer vision,



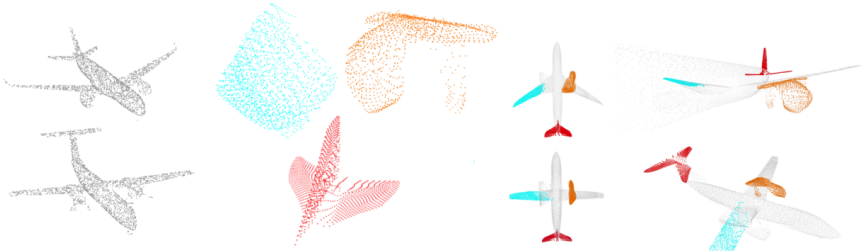
(a) (b) (c) (d)

Fig. 2.8 From point clouds to textured meshes. Starting from a point cloud (a), edges can be added to connect the points in order to form triangles (b). Normals can be computed for these triangles in order to produce shaded renderings (c). Finally, the mesh triangles can be textured, in order to produce shaded and textured renderings (d).



(a) GlobFit (Li et al. (2011))

(b) SuperQuadrics (Paschalidou et al. (2019, 2020))



(c) AtlasNetv2 (Deprelle et al. (2019))

Fig. 2.9 Primitive decompositions for 3D point cloud fitting. Examples of primitive decompositions using different types of primitives. GlobFit uses planes, while Paschalidou et al. (2019, 2020) uses superquadrics and AtlasNetv2 uses learned primitives.

superquadrics were also explored by Paschalidou et al. (2019, 2020) to reconstruct 3D scenes as sets of primitives.

2.2.6 Voxels

Similarly to pixels being a 2D grid discretization of an image, voxels are a 3D grid discretization of a volume. The most commonly used voxels are *occupancy* voxels shown in Figure 2.10, which encode the presence or absence of geometry, e.g. points from a point cloud or triangles from a mesh, with 0 or 1 respectively. However, voxels can be used to encode more general discretization of data in a volume. Other examples include *signed distance fields* (SDF), which is used to encode the *signed* distance to the nearest geometry, where the sign is defined based on the relative position with respect to an oriented surface. Voxelised SDFs were for example used in seminal works KinectFusion (Newcombe et al. (2011)) or ScanNet dataset (Dai et al. (2017a)). Voxels also gained popularity for deep-learning-based 3D computer vision with 3D CNNs, where voxels can now encode general features in a 3D volume used for 3D-aware generative models (Nguyen-Phuoc et al. (2019, 2020)), 3D reconstruction (Liu et al. (2020a); Murez et al. (2020); Peng et al. (2020); Wu et al. (2017, 2018b)), 3D object detection and classification Brock et al. (2016); Maturana and Scherer (2015); Qi et al. (2016); Riegler et al. (2017); Zhou and Tuzel (2018), or novel view synthesis (Henzler et al. (2019); Schwarz et al. (2020); Sitzmann et al. (2019)).

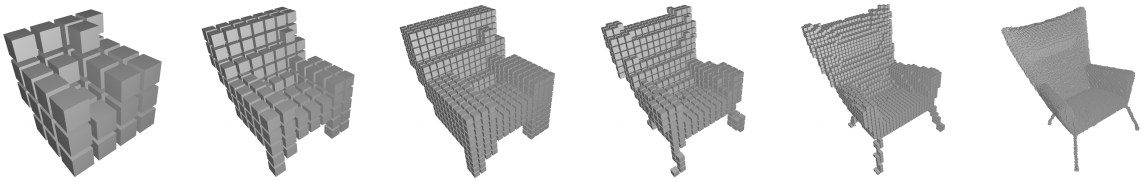
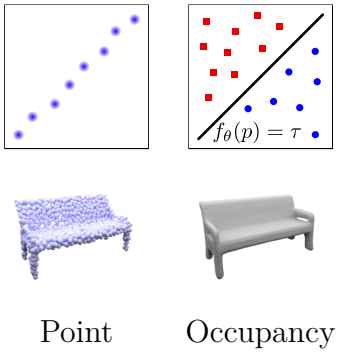


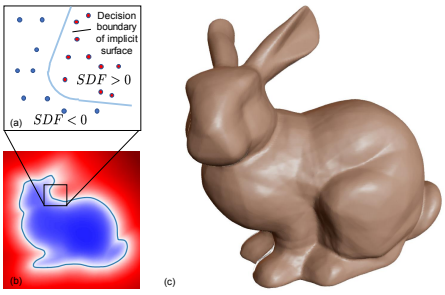
Fig. 2.10 Occupancy voxels at different resolutions. The resolution of the voxel grid increases from left to right, i.e. the voxel size decreases in 3D.

2.2.7 Neural Fields

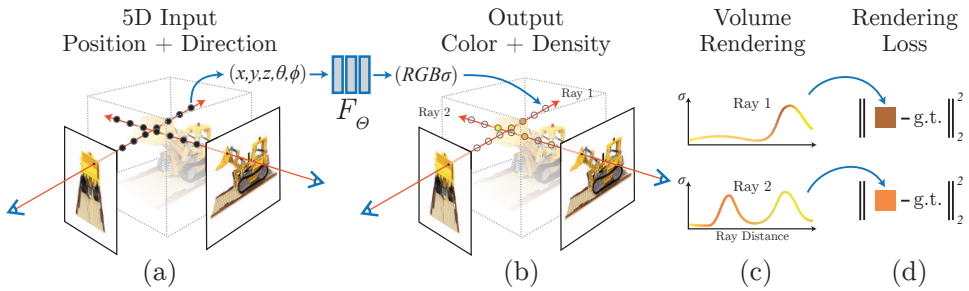
Although we do not make use of neural fields in this thesis, these have become increasingly popular and set new state-of-the-art on multiple 3D benchmarks. In this section we only briefly describe the concept of a neural field, and give a few examples of popular neural fields for 3D computer vision, shown in Figure 2.11. For a comprehensive list of neural fields works see https://neuralfields.cs.brown.edu/.



(a) Occupancy Networks



(b) DeepSDF



(c) NeRF

Fig. 2.11 Overview of popular neural fields in 3D computer vision. Occupancy Network, DeepSDF, and NeRF visualizations were taken from Mescheder et al. (2019), Park et al. (2019), and Mildenhall et al. (2020) respectively.

Definition. Neural fields are fields that are parameterized by neural networks f_θ, which are used to map a spatial and/or temporal coordinate $\boldsymbol{x}$ to a quantity:

$$f_\theta : \boldsymbol{x} \mapsto f_\theta(\boldsymbol{x}). \tag{2.3}$$

Occupancy and Signed Distance Fields. Occupancy Networks Mescheder et al. (2019) learn to represent *occupancy fields* (OF), as they learn to map a 3D coordinate $\boldsymbol{p} = (x, y, z)$ to an occupancy $f_\theta(\boldsymbol{p}) = o \in \{0, 1\}$. Peng et al. (2020) added local 3D features to the Occupancy Networks decoding. Similarly DeepSDF (Park et al. (2019)) learns Signed Distance Fields (SDF), where f_θ maps 3D coordinates to the signed distance $f_\theta(\boldsymbol{p}) = s$ to a 3D surface; if one considers the *inside* of a 3D shape to have negative s, then occupancy fields are equivalent to (minus-)sign of SDFs. Both occupancy and signed distance fields are used as implicit representations for 3D surfaces, where for the former the implicit surface lies on the 3D decision boundary between occupied and non-occupied volume (see Figure 2.11a) and for the latter it lies on the SDF zero-level-set (see Figure 2.11b).

Neural Radiance Fields (NeRF). encode radiance fields, and learn to map 5D coordinates (x, y, z, θ, ϕ), i.e. 3D coordinates and viewing angles, to a color $\boldsymbol{c}$ and geometry density σ. To train NeRFs, Mildenhall et al. (2020) used differentiable volume rendering (Kajiya and Von Herzen (1984)) with ray sampling, such that the weights of the neural network parameterizing the radiance field are optimized to reconstruct a set of posed 2D images (see Figure 2.11c). Using this method, they achieved a new state-of-the-art in novel view synthesis, especially due to the use of high-dimensional embedding of the input spatial coordinates using Fourier Features (Tancik et al. (2020)).

2.3 Learning-Based Single Image Depth Estimation

Single Image Depth Estimation (SIDE) aims to predict depth at all pixels in a single input image. Compared to multi-image scenarios such as multi-view stereo and many Shape-from-X methods, SIDE is highly ill-posed, as no mechanism can solve the ambiguities that arise when interpreting a single-image as a projection of a 3D scene (see Section 1.4). For that reason, most modern methods are data-driven and learning-based, in order to learn a useful prior for depth map inference from a single image.

Therefore, in this section, we only briefly discuss classical Shape-from-X methods in Section 2.3.1. Then, starting from Section 2.3.2, we focus on *learning-based* SIDE by

covering RGB-D datasets that enabled these works. In Section 2.3.3 we discuss methods that use full depth supervision, and finally in Section 2.3.4 we cover methods that do not require full depth supervision.

2.3.1 Shape-from-X

The term *Shape-from-X* broadly encompasses methods that reconstruct 3D shapes given a only single image of a scene, or multiple images captured under different conditions. Many visual cues can be exploited to extract 3D shape from sets of images. Photometric Stereo (Woodham (1980)) exploits shading cues from multiple observations of a scene under varying illumination conditions in order to reconstruct a 3D surface. Shape-from-Shading (Horn and Brooks (1989); Horn (1975)) also exploits shading cues (Barron and Malik (2012a,b, 2015); Frankot and Chellappa (1988); Zhang et al. (1999)) to reconstruct a 3D surface, and can work with a single input image by exploiting geometric priors. Shape-from-Texture (Lobay and Forsyth (2006); Witkin (1981)) exploits deformation of a texture pattern to recover the shape of objects. Shape-from-Defocus (Favaro and Soatto (2005); Subbarao and Surya (1994); Xiong and Shafer (1993)) exploits the relationship between spatial resolution and depth in order to perform depth estimation. While all the above cited methods operate using images of a scene with a static viewpoint, another important family of methods is Structure-from-Motion (SfM). Used alongside Multi-View-Stereo (MVS), these methods can reconstruct 3D scenes by extracting discriminative points and matching them in order to triangulate the 3D scene. We discuss them in more detail in the following.

Structure-from-Motion (SfM) (Schönberger and Frahm (2016); Seitz et al. (2006)) first extracts discriminative keypoints in images using for example classical feature detectors SIFT (Lowe (1999)) or BRIEF (Calonder et al. (2012)), or learned ones Yi et al. (2016). 2D-to-2D keypoint matching is then performed between pairs of images, and these keypoints are then used to estimate the camera poses of each image, and a sparse 3D reconstruction. The SfM pipeline used in the seminal work COLMAP (Schönberger and Frahm (2016); Schönberger et al. (2016)) is shown in Figure 2.12.

Multi-view-Stereo (MVS) relies on multi-image matching to perform multi-view 3D reconstruction. Given a collection of input *posed* images $\{I_k, \mathcal{T}_k\}_{k=1..N}$, the goal of MVS is to match pixels of a reference image I_r to pixels from other images; 3D points can then be obtained from these matches using relative poses between cameras. These poses $\mathcal{T}_k$ must be computed before-hand, therefore most MVS-based reconstructions systems, such as

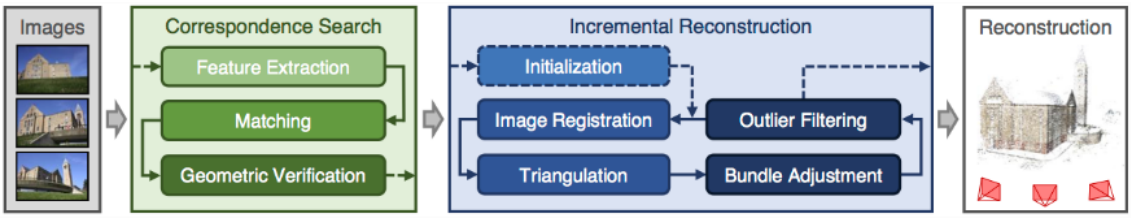


Fig. 2.12 Structure-from-Motion pipeline from COLMAP by Schönberger and Frahm (2016)

the seminal work COLMAP by Schönberger and Frahm (2016); Schönberger et al. (2016) rely on a SfM algorithm. Finally, MVS systems are able to provide semi-dense depth maps aligned with each input image, as well as a metric and semi-dense 3D reconstructions. For a tutorial on classical MVS, see Furukawa et al. (2015). Several deep learning based methods have also emerged to perform MVS (Huang et al. (2018a); Yao et al. (2018)), and became state-of-the-art for the task, however classical methods such as COLMAP still remain competitive, as studied by Darmon et al. (2021). NeRF-based (Mildenhall et al. (2020)) methods, discussed in Section 2.2 have also gained increasing popularity to perform MVS 3D reconstruction (Wei et al. (2021)).

2.3.2 RGB-D Datasets

As discussed in Section 2.1, datasets are a key component to enable deep learning based methods to work, as they require large amounts of training data. In this section we cover some of the most popular RGB-D datasets used in the litterature for deep learning based monocular depth estimation. Although we separate these datasets into four categories, time-of-flight (Section 2.3.2.1), structured-light (Section 2.3.2.2), structure-from-motion (Section 2.3.2.3) and synthetic (Section 2.3.2.4), all RGB-D datasets are treated as sample pairs of aligned RGB and depth images. While section covers some of the most popular RGB-D datasets used depth estimation litterature, it is not an exhaustive list. We refer the reader to Firman (2016) and Lopes et al. (2022) for surveys on RGB-D datasets.

2.3.2.1 Time-of-flight

Time-of-flight based sensors acquire depth based on a simple principle: the measurement of time required for a laser or radar wave pulse/burst to travel from the sensor to the scene then back, after reflection onto the scene.

Make3D The Make3D dataset (Saxena et al. (2006); Saxena et al. (2009)) contains 534 outdoor high-resolution images, 400 for training, and 134 for testing, and are paired with small resolution ground truth. Depth is acquired using a custom laser-based 3D scanner mounted on a robot.

KITTI The KITTI Vision Benchmark Suite Geiger et al. (2013, 2012) has been designed to train and evaluate computer vision methods with autonomous driving applications. While this dataset is used for multiple tasks such as optical or scene flow estimation, visual odometry, 3D object detection or semantic segmentation, we focus on the depth estimation benchmark. The dataset consists of 22,600 calibrated stereo video pairs captured by a car around a Karlsruhe, a city in Germany. In order to acquire ground truth depth, this car is also equipped with a 360 degrees LIDAR, which produces a sparse 3D point cloud reconstruction. We use the KITTI dataset both for training and evaluation of WaveletMonoDepth as described in Chapter 5.

iBims The iBims dataset is a moderate-size but high-quality RGB-D dataset used for evaluation of single-image depth estimation methods. Compared to its other datasets, iBims uses a custom laser-based acquisition setup that has very low depth noise, sharp depth edges, and no occlusions. We used iBims to evaluate our first two monocular depth estimation methods described in Chapters 3-4.

Other notable time-of-flight datasets used in the litterature but not used in this thesis include the Matterport dataset (Chang et al. (2017)) or the DIODE dataset (Vasiljevic et al. (2019)).

2.3.2.2 Structured-light

Structured-light sensors rely on projection of a 2D pattern on the 3D scene, and measuring disparity by matching known parts of this pattern after they are deformed by the 3D scene (see Figure 1.7). They are usually performed using infrared projectors and receivers. For a tutorial on structured-light for 3D reconstruction, we refer to Geng (2011).

NYUv2 Depth Introduced by Silberman et al. (2012), the NYUv2-Depth dataset contains 464 indoor video scenes captured with a Microsoft Kinect sensor, a structured-light sensor that enables depth sensing at moderate ranges with aligned RGB images. From the 249 training scenes, 120K pairs of RGB and depth images can be used for training. This dataset has arguably been a cornerstone to enable tremendous progress in

deep learning based monocular depth estimation, pioneered by Eigen and Fergus (2015); Eigen et al. (2014). NYUv2 Depth is used both for training and evaluation in all our monocular depth estimation works, described in Chapters 3-4-5.

ScanNet (Dai et al. (2017a)) Similarly to NYUv2, ScanNet is an indoor dataset collected using a Microsoft Kinect sensor. In addition to RGB-D sequences of 2.5 million images obtained from more than 1500 scenes, ScanNet provides a 3D scene reconstruction for each scene, using a 3D mesh representation that we use in Chapter 6.

2.3.2.3 Multi-View Stereo

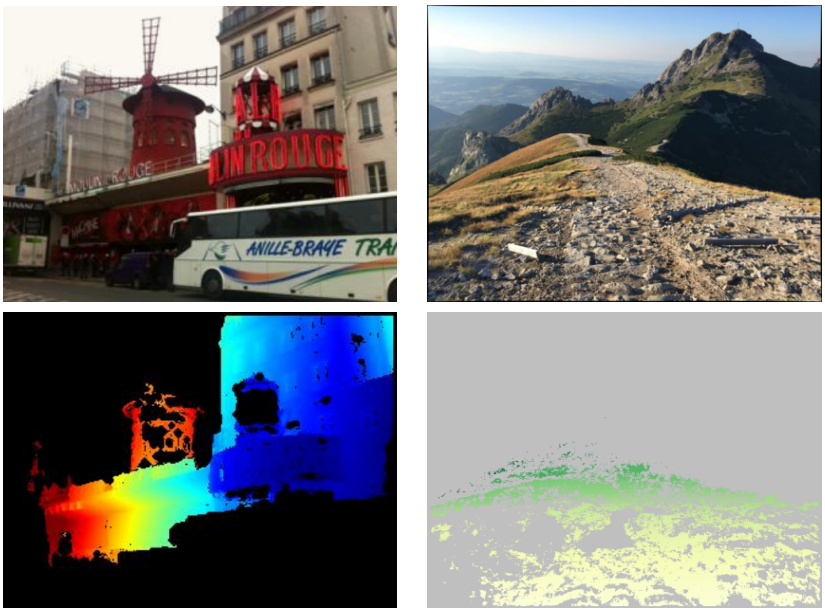
As discussed in Section 2.3.1, Multi-View Stereo (MVS) consists in performing dense 3D reconstruction via triangulation using multiple cameras viewing the same scene from different viewpoints by matching pixels between views. Contrary to structured-light or time-of-flight sensors, MVS can perform *passive* reconstruction, as the 3D reconstruction system does not rely on projecting waves onto the 3D scene. Such passive reconstructions systems can be particularly suitable for surveillance or military applications, where it is desirable to perform remote reconstruction while remaining undetected. As mentioned in Section 2.3.1, 3D reconstruction using MVS requires knowledge of relative camera poses, which usually performed comes from a first Structure-from-Motion step. In this section we only cover two large scale "in the wild" photogrammetry RGB-D datasets, shown in Figure 2.13 but refer the reader to Lopes et al. (2022) for a survey on MVS datasets.

MegaDepth (Li and Snavely (2018)) is a popular large scale outdoor RGB-D dataset, which comprises 130,000 samples generated using COLMAP Schönberger and Frahm (2016); Schönberger et al. (2016), an method that combines SfM and MVS to generate semi-dense depth maps.

Mapillary Planet Scale Dataset (MPSD) (Antequera et al. (2020)) MPSD contains 750,000 RGB-D image pairs, collected from 50,000 scenes. These images were also collected in a broader set of countries and environments than its predecessors, as MegaDepth for example collected images from 200 scenes.

2.3.2.4 Synthetic datasets

Synthetic datasets for 3D scene understanding have become popular after deep learning breakthroughs in depth estimation (Eigen et al. (2014); Laina et al. (2016)), semantic segmentation (He et al. (2017a); Zhao et al. (2017)) and object detection (Redmon

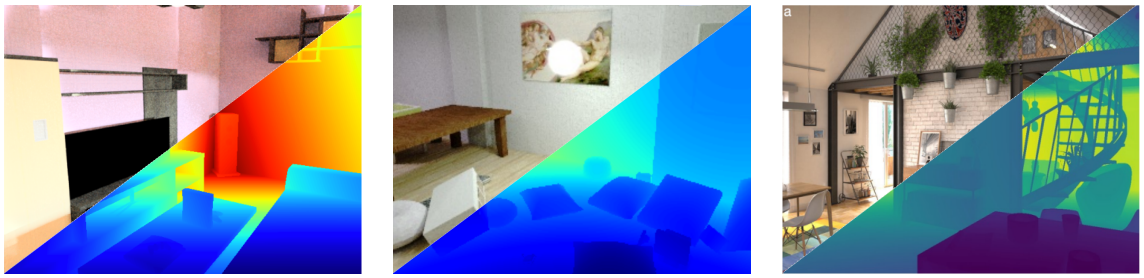


(a) Megadepth (b) MPSD

Fig. 2.13 Large scale multi-view stereo datasets.

et al. (2016); Ren et al. (2015)). Since they are generated by rendering 3D scenes using computer graphics, their geometry annotations are noise free. However, synthetically generated datasets often exhibit a large domain gap with datasets acquired with real cameras. To solve this issue, two types of methods are used when generating synthetic datasets: (i) photo-realistic rendering, which use the latest advances in physically-based rendering (PBR), but often result in either noisy or abnormally perfect images, or (ii) domain randomization, which generates fuzzy datasets with random textures, object and camera poses, etc. While the first aims to reduce simulation-to-reality domain gap, the latter rather aims to improve generalization capabilities of neural networks. While other synthetic datasets are listed in the survey by Lopes et al. (2022), we present three datasets, shown in Figure 2.14.

SUNCG (Song et al. (2017)) The SUNCG dataset was the first to introduce computer graphics to generate a dataset of indoor scenes suitable for 3D scene understanding purposes, as it contained 45,622 3D models of indoor scenes. Photorealistic renderings have later been used to generate image datasets, such as PBRS Zhang et al. (2017) which we use in our work SharpNet, described in Chapter 3. However, although it was widely



(a) SUNCG (PBRS rendering by Zhang et al. (2017))

(b) Scene-RGBD

(c) Hypersim

Fig. 2.14 RGB-D pairs of popular synthetic datasets.

used by the computer vision community, the SUNCG dataset became unavailable for research, and its derivative datasets such as PBRS also suffered the same fate.[1]

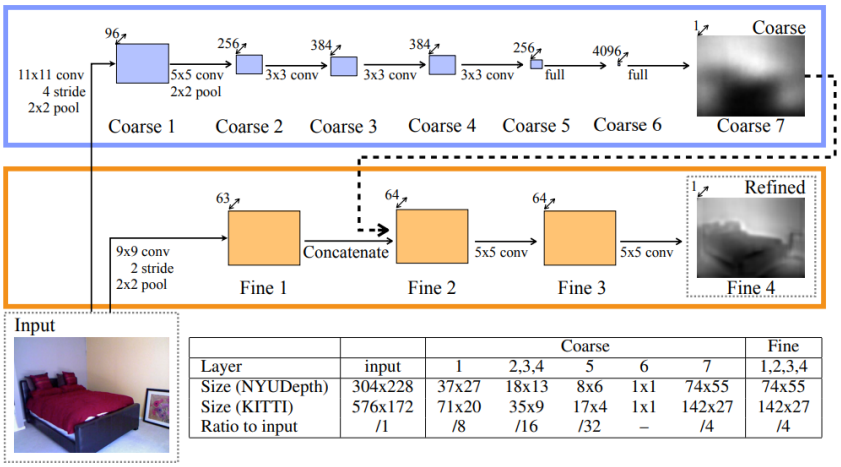
Scene-RGBD (McCormac et al. (2017)) is a dataset of 5 million rendered RGB-D images from over 15K trajectories in synthetic layouts with random but physically simulated object poses. Compared to SUNCG which provides a discrete set of poses, Scene-RGBD uses realistic camera trajectories, enabling sequential applications such as SLAM.

Hypersim (Roberts et al. (2021)) is a photorealistic synthetic dataset for holistic indoor scene understanding. Hypersim leverages a large repository of synthetic scenes created by professional artists, and comprises of 77,400 rendered images of 461 indoor scenes with detailed per-pixel labels and corresponding ground truth geometry.

2.3.3 Fully Supervised Methods

Saxena et al. (2009) pioneered supervised learning for SIDE. They used a Markov Random Field (MRF) model that incorporates multi-scale local and global image features using super-pixels. Shortly after the first breakthroughs in deep learning for computer vision (Krizhevsky et al. (2012)) and the release of the NYUv2-Depth (Silberman et al. (2012)) dataset, deep learning based methods took over the state-of-the-art in monocular depth estimation. In this section we cover several of these deep learning based methods; there also exist many surveys aiming to provide comprehensive reviews of the field (Ming et al. (2021); Zhao et al. (2020)).

[1] https://github.com/DLR-RM/BlenderProc/issues/11



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Fig. 2.15 Deep Neural Network architecture from Eigen et al. (2014).

Multi-scale architectures have been employed since the seminal work by Eigen et al. (2014). They typically use a cascade of convolutions and down-sampling operations such as pooling or strided convolutions to provide a low-resolution representation at the bottleneck which aggregates global features from the input image. This low-resolution representation, is then progressively upsampled using convolutions which locally refine the initial low-resolution representation, and ultimately produce a depth map at the original image resolution. As seen in Figure 2.15, while in Eigen et al. (2014) this low-resolution representation is directly a depth map, Laina et al. (2016) introduced deeper architecture where the bottleneck representation is high-dimensional. Because of its success for many other computer vision tasks, the U-Net (Ronneberger et al. (2015)) architecture has also been used as a basis in many MDE works (Godard et al. (2019); Ramamonjisoa and Lepetit (2019); Yin et al. (2019)). Recently, Ranftl et al. (2021) used Vision Transformers (ViT) (Dosovitskiy et al. (2020)) as an encoder in an encoder-decoder architecture with bottleneck, similarly to U-Net. As shown in Figure 2.16 it also performs down-sampling down to a 1/32 scale, then progressively refines and up-samples feature maps up to the initial resolution using a residual convolutional up-sampling decoder with skip connections.

Regression with classification has become a common paradigm for MDE after the seminal work DORN by Fu et al. (2018) was first introduced. Fu et al. (2018) reformulate regression-based MDE as a classification problem, by first dividing the full continuous range of depths into discrete bins, then training the network as a pixel-wise classifier that learns to classify a pixel into one of these bins. AdaBins (Bhat et al. (2021)) extends DORN by adding adaptivity in bins; they design a hybrid regression-classification

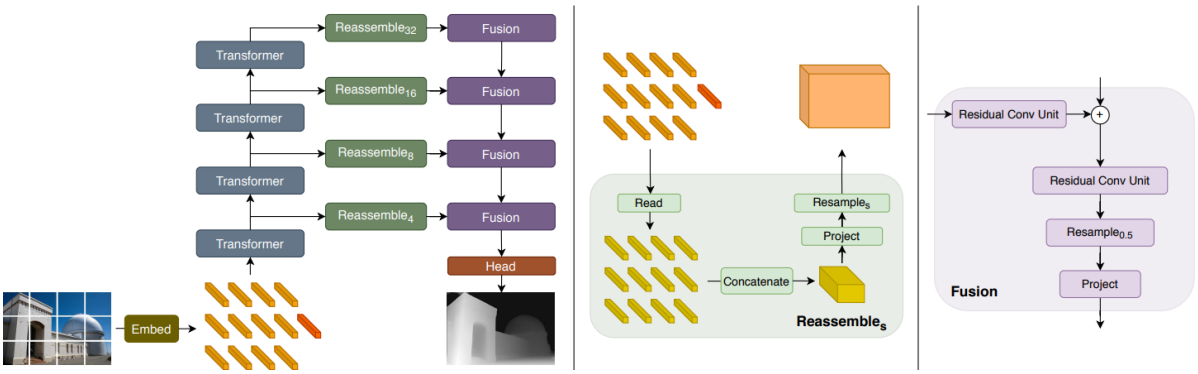


Fig. 2.16 Architecture of DPT by Ranftl et al. (2021).

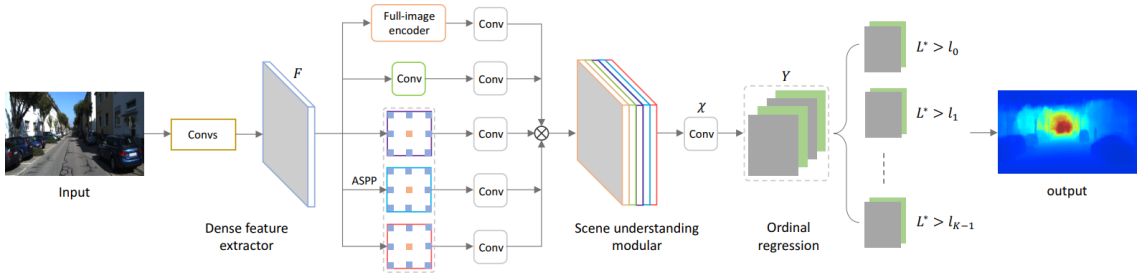


Fig. 2.17 Deep Neural Network architecture from Fu et al. (2018).

network that predicts the depth bins from the input image, then classify each pixel into these bins. However, contrary to DORN, they combine per-bin scores using softmax to produce depth predictions as a weighted sum over bins, enabling the use of regular depth estimation losses.

Pair-wise ranking losses have also been explored after their introduction to MDE by Chen et al. (2016). The core idea of the work is presented in Figure 2.18. While their network predicts dense metric depth maps, the loss is computed using sparse pair-wise relative depth annotations. Pairs of pixels are annotated with +1 if the first point is the closest, -1 if it is furthest, and 0 if they are at similar depth. The network is then trained using an objective function that favors large depth differences between annotated points if pairs have different depth (-1 or +1), and small differences if they have similar depth (0). This objective function, used alongside their large scale Depth In the Wild (DIW) dataset proved sufficient to achieve monocular depth estimation without dense depth supervision. Xian et al. (2020) improve the ranking loss proposed by Chen et al. (2016) with structure-guided sampling, such that pairs of points close to depth edges are more frequently sampled when computing the ranking loss.

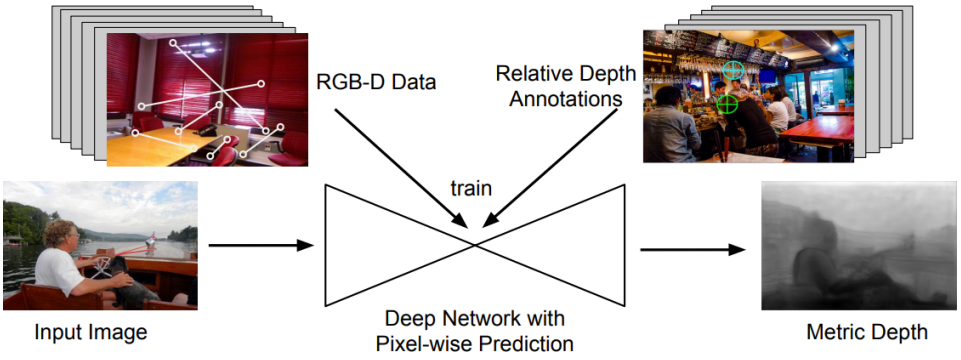


Fig. 2.18 Overview of Depth In the Wild by Chen et al. (2016).

Scale-invariant losses were first designed by Eigen et al. (2014) to train MDE networks on RGB-D datasets which have large depth ranges. Their loss, in Equation (2.4) ensures predictions are penalized equally regardless of the ground truth scene scale s:

$$\ell(b\hat{D}, \boldsymbol{D}, s) = \log(s \cdot \hat{\boldsymbol{D}}) - \log(s \cdot \boldsymbol{D}) = \log(\hat{\boldsymbol{D}}) - \log(\boldsymbol{D}), \tag{2.4}$$

where $\hat{\boldsymbol{D}}$ and $\boldsymbol{D}$ are the predicted and ground truth depth respectively. When computed in log-scale, the loss no longer depends on the scale s.

Furthermore, generalization is a highly desirable feature for MDE systems; training using diverse datasets is a sensible option to produce more robust depth estimation systems. However, training with multiple datasets presents some challenge, because ground truth can come in many forms, such as dense or sparse depth maps, scale-less data from SfM, or disparity maps. To this end, Ranftl et al. (2020) designed scale and shift invariant losses, and showed that training with diverse datasets using their losses exhibit superior zero-shot generalization capabilities on new test datasets than state-of-the-art. Yin et al. (2021) also propose to learn affine-invariant depth using an heterogeneous loss training strategy based on undistorting the 3D point clouds generated from the predicted depth maps (see Figure 2.19) using scale and shift prediction.

Auxiliary tasks have been exploited for MDE in order to incorporate extra supervision to the main task of depth estimation. Most of these methods originate from the availability of labels for other tasks on the same RGB-D dataset.

Surface normal estimation has been widely explored jointly with depth estimation, since as surface normals represent local surface orientations, they can be computed from depth maps, which also represent a 3D surface. Qi et al. (2018) performs joint depth and surface normal estimation from a single image in two steps, shown in Figure 2.20. They first predict initial depth and normal estimates, then refine both using closed-form

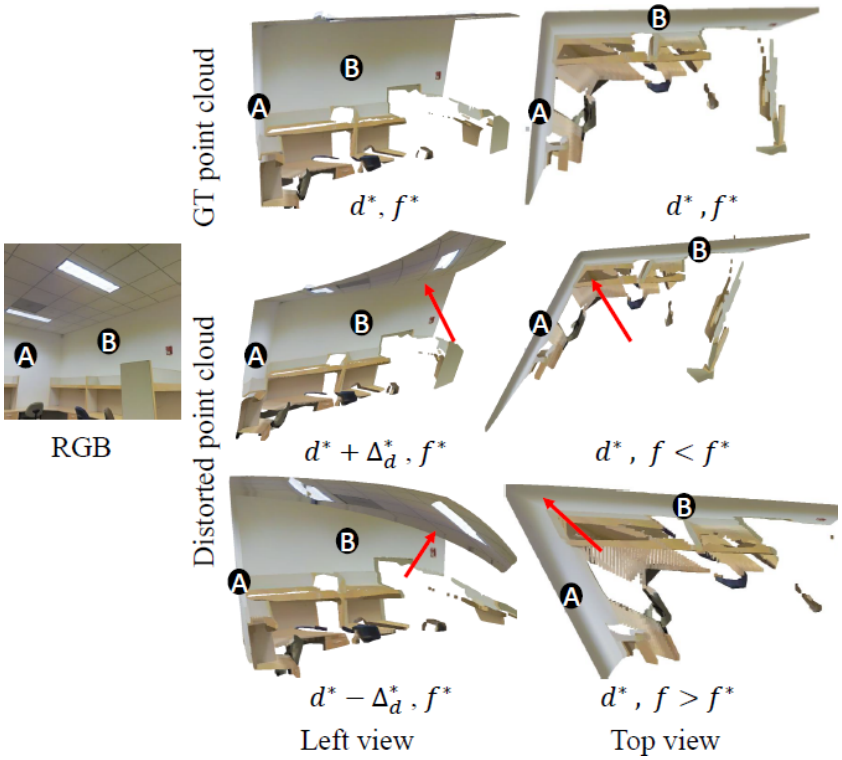


Fig. 2.19 3D distortion induced by scale and shift errors. Predicting depth maps with wrong scale and shift produces distorted 3D reconstructions. Yin et al. (2021) propose to separately predict affine-invariant then depth and scale and shift to improve monocular depth estimation methods. Figure from Wei Yin's PhD thesis.

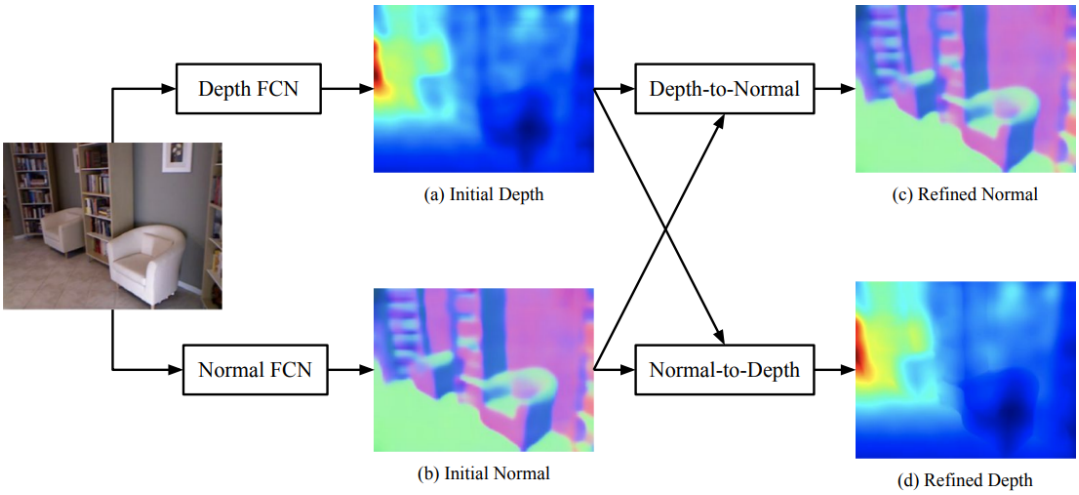


Fig. 2.20 GeoNet (Qi et al. (2018)) pipeline overview.

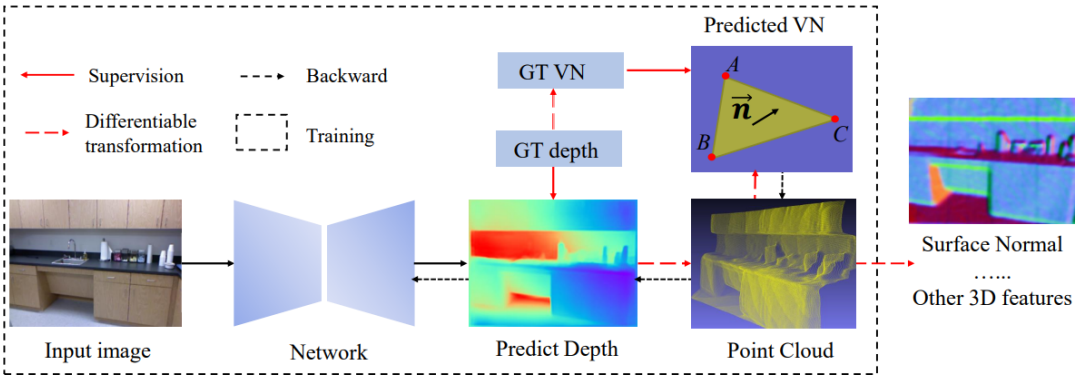


Fig. 2.21 VNL (Yin et al. (2019)) pipeline overview.

geometric relationships between the two modalities. Yin et al. (2019) also exploit the relationship between depth and normals to enforce constraints on depth estimates in 3D space, as shown in Figure 2.21. Finally our method SharpNet, detailed in Chapter 3 also exploits geometry constraints between depth and normals but also occlusion boundaries.

Semantic segmentation has also been used extensively, largely due to the availability of semantic labels in some popular RGB-D datasets, such as NYUv2 (Silberman et al. (2012)). Eigen and Fergus (2015) pioneered joint semantic segmentation and depth estimation on NYUv2. Wang et al. (2015) aimed to build a unified framework for joint semantic segmentation and depth estimation by grouping semantic classes. Jiao et al. (2018) (see Figure 2.22) leverage semantic annotations to use semantically-guided attention in depth estimation. Wang et al. (2020) explored a three step depth estimation pipeline shown in Figure 2.23, which first performs instance and semantic segmentation,

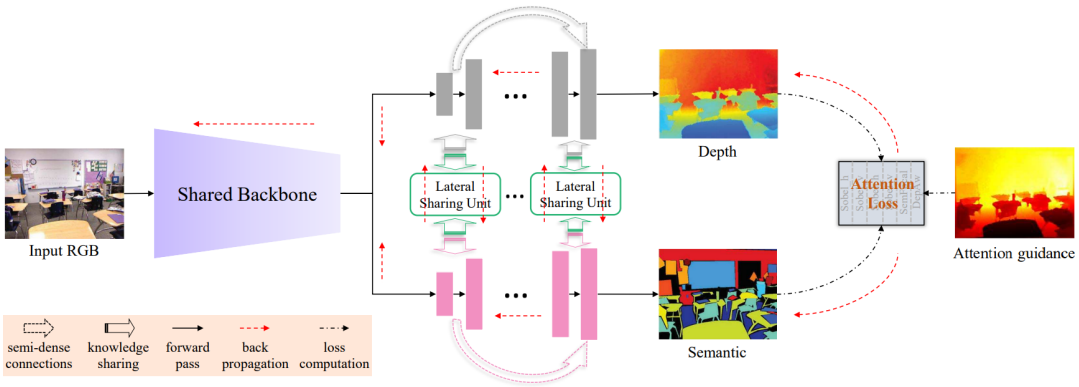


Fig. 2.22 Look Deeper into Depth (Jiao et al. (2018)) pipeline overview.

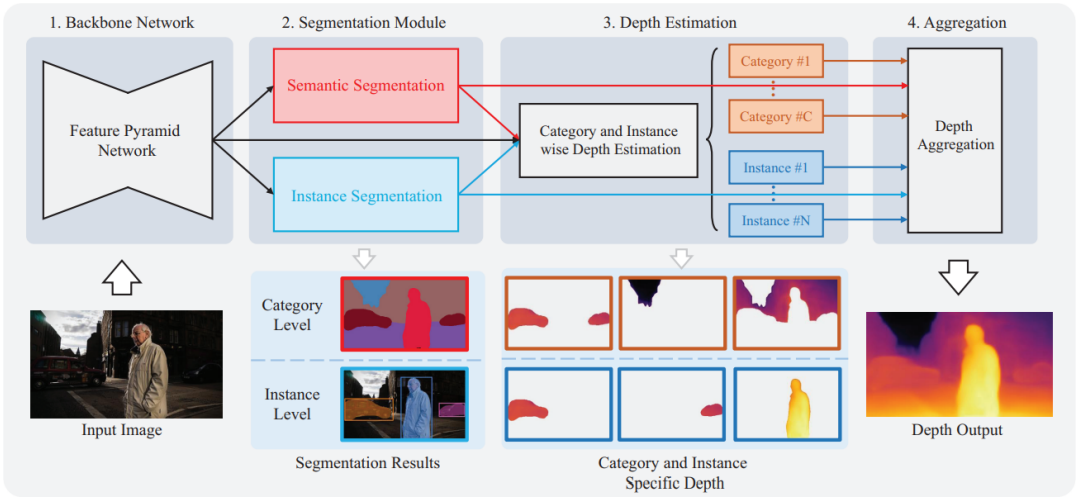


Fig. 2.23 SDC-Depth (Wang et al. (2020)) pipeline overview.

then performs separate category-based and instance-based depth estimation, then finally aggregates all depth estimates into a unified depth output.

2.3.4 Self-Supervised Methods

While fully supervised MDE methods remain the most accurate ones, another line of works aims to alleviate the need for full depth supervision by exploiting consistency between consecutive temporal frames or stereo frames.

Depth estimation as view synthesis by Zhou et al. (2017). Zhou et al. (2017) introduced the first monocular self-supervised approach presented in Figure 2.24, which trains a single-image depth estimation network jointly with a relative pose estimation network. Their training loss is formulated as a cross-view photometric reconstruction loss,

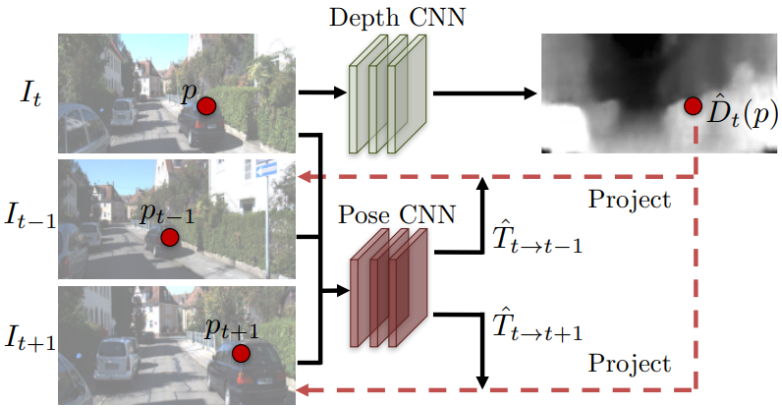


Fig. 2.24 Overview of the depth estimation as view synthesis framework proposed by Zhou et al. (2017).

a loss commonly use in view-synthesis. Let I_t and I_s be two nearby frames, where I_t is a target view and I_s a source view. Given a predicted depth $\hat{\boldsymbol{D}}_t$ from I_t and a predicted 4x4 transformation matrix $\hat{\mathcal{T}}_{t\to s}$, the goal is to perform *inverse* warping using Spatial Transformer Networks (STN) (Jaderberg et al. (2015)), by sampling the source frame I_s in order to reconstruct the *input* target image I_t. Let $I_{s\to t}$ denote the synthesized target image from sampling I_s. The loss then becomes:

$$\ell_p = pe(I_t, I_{s\to t}), \tag{2.5}$$

where $pe()$ is a photometric error loss that measures the deviation between the target image and its reconstruction via inverse warping. Most of the works in the litterature use a combination of L1 and Structural Similarity (SSIM) losses to compute photometric error.

In order to compute $I_{s\to t}$, the grid sampling coordinates p_t of the target must be transformed into the source's I_s coordinates p_s:

$$\begin{aligned} \hat{p_s} &= \text{proj}(\hat{\boldsymbol{D}}_t, \hat{\mathcal{T}}_{t\to s}, p_t) \\ &\sim K\hat{\mathcal{T}}_{t\to s}\hat{\boldsymbol{D}}_t(p_t)K^{-1}p_t, \end{aligned} \tag{2.6}$$

where K is the known intrinsics matrix. Note that this equation uses the back-projection Equation (2.2) described in Section 2.2.2, where $(X, Y, Z)^T = \hat{\boldsymbol{D}}_t(p_t)K^{-1}p_t$ are the 3D coordinates of a point obtained from the depth value at camera coordinate $p_t = (x, y, 1)^T$. Because $\hat{p_s}$ in Equation (2.6) is continuous, the sampling from I_s is performed using bilinear interpolation, such that:

$$I_{s\to t}(p_t) = I_s\langle\hat{p_s}\rangle, \tag{2.7}$$

where $\langle\rangle$ is the bilinear interpolation sampling operator.

Note that while most self-supervised use the inverse warping, few works have performed *soft* forward warping to warp the target (input) image onto the source (nearby view) image (GonzalezBello and Kim (2020); Xie et al. (2016)).

Loss masking. The loss in Equation (2.5) can finally be computed. Zhou et al. (2017) use a weighted L1 loss, where the weight is obtained by a network trained to predict *explanability masks* in order to solve cases which violate view synthesis assumptions:

- the scene is static, and motion is only explained by ego-motion,
- there are no occlusion/disocclusion between target and source views,
- the surface of objects is lambertian, such that photometric reconstruction losses are meaningful

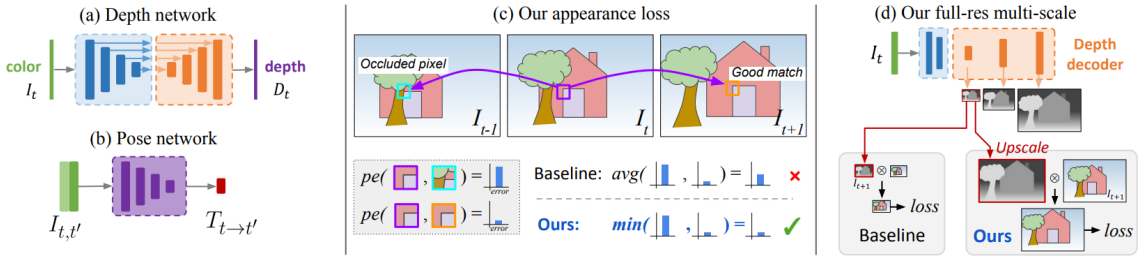


Fig. 2.25 Overview of Monodepth2 (Godard et al. (2019)) architecture.

To solve the occlusion/disocclusion issue, in Monodepth2 Godard et al. (2019) reformulate the reprojection error as a minimum reprojection loss. As shown in Figure 2.25, they alleviate the problem of sampling nearby source images where pixels of the target image are *not visible*, which mostly happens due to occlusion and ego-motion. The minimum reprojection loss effectively selects the best nearby view to sample from for each pixel. Doing so, they obtain more sharp and accurate depth maps than state-of-the-art. Previous work (Luo et al. (2019); Vijayanarasimhan et al. (2017)) also used predicted masks to handle static scenes, moving objects, and occlusion issues. While Monodepth2 also uses a mask, they use simple binary automatic mask to compute the loss, which is only activated for pixels where inverse-warping leads to better (lower) photometric error *pe* than when not applying warping.

Exploiting Stereo Several previous work extended the view-synthesis based unsupervised depth estimation method by Zhou et al. (2017) by adding multi-view consistency. Monodepth (Godard et al. (2017)) exploits left-right consistency at training time since

the KITTI dataset (Geiger et al. (2013)) contains *stereo* videos. MonoResMatch (Tosi et al. (2019)) extends this approach by distilling stereo knowledge to the monocular approach, adding a proxy loss that leverages labels obtained via Semi-Global-Matching (SGM) (Hirschmuller (2005, 2007)). Watson et al. (2019) further extend MonoResMatch with Depth Hints, shown in Figure 2.26 in order to better exploit SGM labels by leveraging a hybrid reprojection loss which selects the best reprojection between SGM-based reprojection and the standard reprojection using Equation (2.6). Poggi et al. (2018b) exploits trinocular assumptions. We recommend the survey by Poggi et al. (2021) for a study on synergies between binocular stereo and monocular depth estimation.

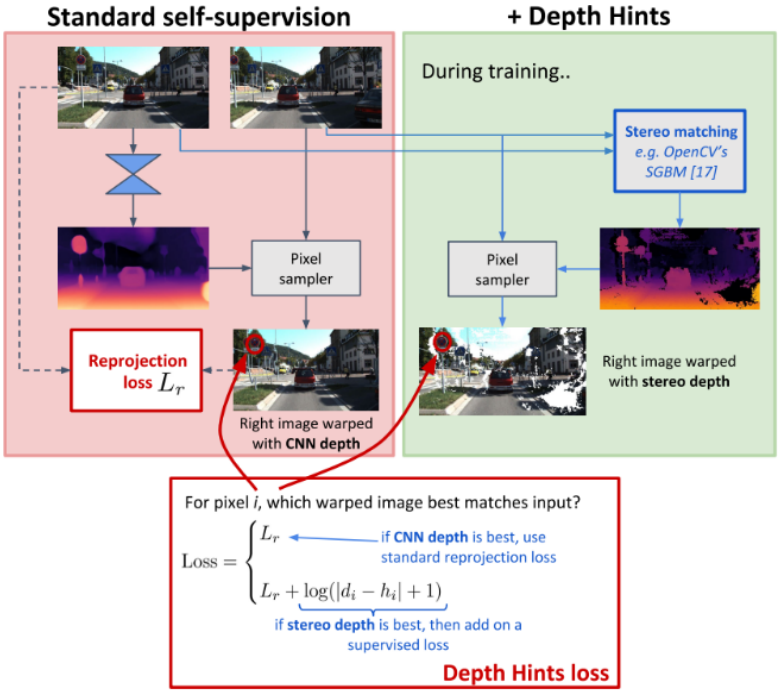


Fig. 2.26 Overview of Depth Hints by Watson et al. (2019). This figure is taken from Jamie Watson's slides at the CVPR2020 tutorial on MDE, which we highly recommend.

Discriminative losses were also explored (Aleotti et al. (2018); CS Kumar et al. (2018); Groenendijk et al. (2020)) to improve the quality of images during inverse warping, and thus improve the depth maps. The general frame work is shown in Figure 2.27.

Auxiliary tasks were used extensively to guide depth estimation. Geometry estimation tasks such as edge and normal estimation (Yang et al. (2018b,c)), or optical flow (Chen et al. (2019c); Mahjourian et al. (2018); Ranjan et al. (2019); Yin and Shi (2018)). Semantic segmentation was also explored as a auxiliary task for depth estimation to obtain better depth edges (Zama Ramirez et al. (2018); Zhu et al. (2020)), solve moving

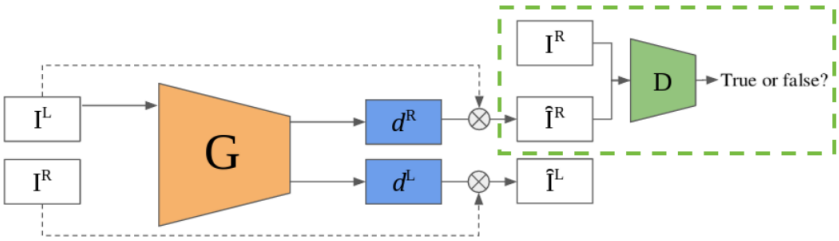


Fig. 2.27 General architecture of self-supervised monocular depth estimation architectures with GANs (Figure by Groenendijk et al. (2020)).

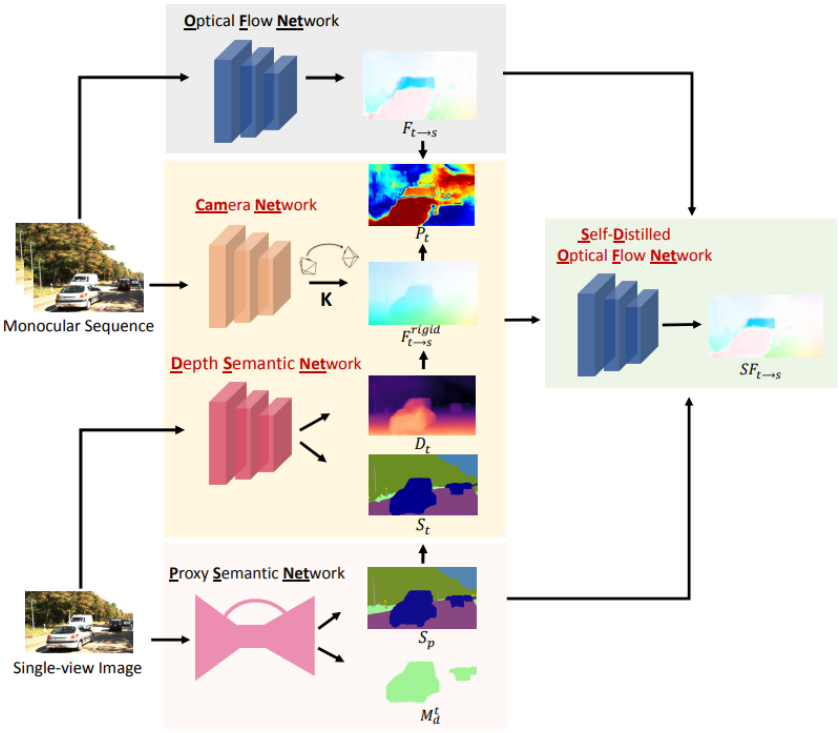


Fig. 2.28 Architecture overview of ΩNet by Tosi et al. (2020).

objects that lead to ambiguities in self-supervised depth estimation (Klingner et al. (2020)), or complement other geometry estimation tasks Tosi et al. (2020) (see Figure 2.28).

Chapter 3

Predicting Sharp Occlusion Boundaries in Monocular Depth Estimation with Geometry Guidance

The work described in this chapter is based on the following publication:
SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation, Michaël Ramamonjisoa and Vincent Lepetit, published in the *3D Reconstruction in the Wild workshop* at ICCV 2019.

Abstract

We introduce SharpNet, a method that predicts an accurate depth map for an input color image, with a particular attention to the reconstruction of occluding contours: Occluding contours are an important cue for object recognition, and for realistic integration of virtual objects in Augmented Reality, but they are also notoriously difficult to reconstruct accurately. For example, they are a challenge for stereo-based reconstruction methods, as points around an occluding contour are visible in only one image. Inspired by recent methods that introduce normal estimation to improve depth prediction, we introduce a novel term that constrains depth and occluding contours predictions. Since ground truth depth is difficult to obtain with pixel-perfect accuracy along occluding contours, we use synthetic images for training, followed by fine-tuning on real data. We demonstrate our approach on the challenging NYUv2-Depth dataset, and show that our method outperforms the state-of-the-art along occluding contours, while performing on par with the best recent methods for the rest of the images. Its accuracy along the occluding

contours is actually better than the 'ground truth' acquired by a depth camera based on structured light. We show this by introducing a new benchmark based on NYUv2-Depth for evaluating occluding contours in monocular reconstruction, which is our second contribution. Project Code is available at https://michaelramamonjisoa.github.io/projects/SharpNet

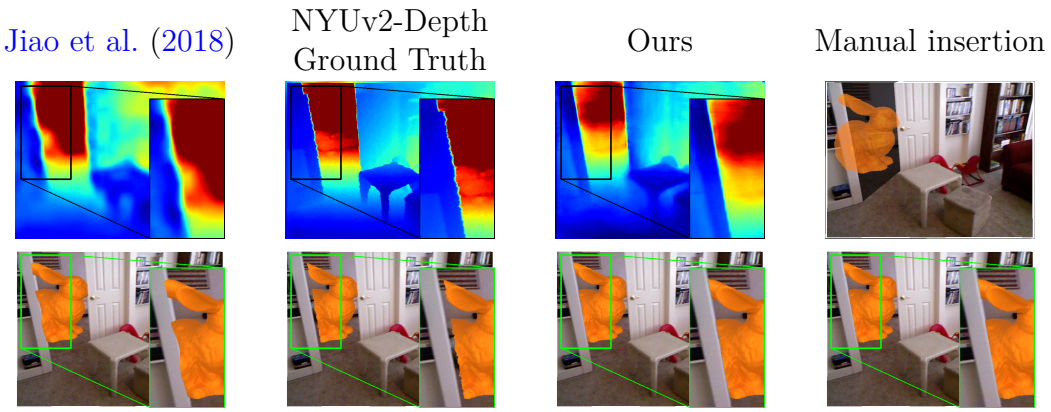


Fig. 3.1 Our SharpNet method shows significant improvement over state-of-the-art monocular depth estimation methods in terms of occluding contours accuracy, and even produces sharper edges along these contours than structured-light depth cameras. In this figure we augment an RGB image from NYUv2 (Silberman et al. (2012)) with a virtual Stanford rabbit using different depth maps for occlusion-aware integration. The first three rows show the depth map used for occlusion-aware insertion (top) and resulting augmentation (bottom). An error of only a few pixels along occluding contours can significantly degrade the realism of the integration, comparatively to manual insertion (last column) using a binary mask.

3.1 Introduction

As discussed in Section 2.3, Monocular Depth Estimation (MDE) is highly ill-posed, therefore most methods rely on data-driven approaches, most specifically employing deep neural networks.

Despite recent advances in monocular depth estimation, occluding contours remain difficult to reconstruct correctly from depth, while they are still an important cue for object recognition, and for augmented reality or path planning, for example. This has several causes: First, the depth annotations of training images are likely to be inaccurate along the occluding contours, if the depth annotations are obtained with a stereo reconstruction method or a structured light camera. This is for example the case for the NYUv2-Depth dataset (Silberman et al. (2012)), which is an important benchmark used by many recent works for evaluation. This is because on one or both sides of the occluding contours lie 3D points that are visible in only one image, which challenges the 3D reconstruction (Szeliski (2011)). Structured light cameras essentially rely on stereo reconstruction, where one image is replaced by a known pattern (Han et al. (2013)), and therefore suffer from the same problem. Secondly, occluding contours, despite their importance, represent a small part of the images, and may not influence the loss function used during training if they are not handled with special care.

In this paper, we show that it is possible to learn to reconstruct occluding contours more accurately by adding a simple term that constrains the depth predictions together with the occluding contours during learning. This approach is inspired by recent works that predict the depths and normals for an input image, and enforce constraints between them (Qi et al. (2018); Wang et al. (2016); Yang et al. (2018b)). A similar constraint between depth and occluding contours can be introduced, and we show that this results in better reconstructions along the occluding contours, without degrading the accuracy of the rest of the reconstruction.

Specifically, we train a network to predict depths, normals, and occluding contours for an input image, by minimizing a loss function that integrates constraints between the depths and the occluding contours, and also between depths and normals. We show that these two constraints can be integrated in a very similar way with simple terms in the loss function. At run-time, we can predict only the depth values, making our method suitable for real-time applications since it runs at 150 fps on 640×480 images.

We show that each aspect of our training procedure improves the depth output. In particular, our experiments show that the constraint between depths and occluding contours is important, and that the improvement is not only due to multi-task learning.

Learning to predict the normals in addition to the depths and the occluding contours helps the convergence of training towards good depth predictions.

We demonstrate our approach on the NYUv2-Depth dataset, in order to compare it against previous methods. Since we need training data with pixel perfect depth annotation along the occluding contours, we use synthetic images to pretrain the network before fine-tuning on NYUv2-Depth. We simply use the object instance boundaries given by the synthetic dataset as training annotations of the occluding contours. However, we only use the depth ground truth as training data when finetuning on the NYUv2-Depth dataset.

A proper evaluation of the accuracy of the occluding contours is difficult. Since the "ground truth" depth data is typically noisy along occluding contours, as in NYUv2-Depth, an evaluation based on this data would not be representative of the actual quality. Even with better depth data, identifying occluding contours automatically as ground truth depth discontinuities would be sensitive to the parameters used by the identification method (Canny (1986)) (see Fig. 3.4).

We therefore decided to annotate manually the occluding contours in a subset of 100 images randomly sampled from the NYUv2-Depth validation set, which we call the NYUv2-OC dataset. Our annotations and our code for the evaluation of the occluding contours are publicly available for comparison. We evaluate our method on this data in terms of 2D localization, in addition to evaluating depth estimation on the NYUv2-Depth validation set using more standard depth estimation metrics (Eigen and Fergus (2015); Eigen et al. (2014); Laina et al. (2016)). Our experiments show that while achieving competitive results on the NYUv2-Depth benchmark by placing second on all of them, we outperform all previous methods in terms of occluding contours 2D localization, especially the current leading method on monocular depth estimation (Jiao et al. (2018)).

3.2 Related Work

Monocular depth estimation (MDE) from images made significant progress recently. Most works now employ deep neural network and large datasets in order to solve the ill-posedness of MDE. While we discussed general related work in Section 2.3, in the following, we focus the discussion on the quality of depth edges.

Supervised Monocular Depth Estimation With the development of large datasets of images annotated with depth data (Chang et al. (2017); Geiger et al. (2013); McCormac et al. (2017); Silberman et al. (2012); Song et al. (2015); Zhang et al. (2017)),

many supervised methods have been proposed. Eigen and Fergus (2015); Eigen et al. (2014) used multi-scale depth estimation to capture global and local information to help depth prediction. Given the remarkable performances they achieved on both popular benchmarks NYUv2-Depth (Silberman et al. (2012)) and KITTI (Geiger et al. (2013)), more work extended this multi-scale approach (Li et al. (2017); Xu et al. (2017)). Previous work also considered ordinal depth classification Fu et al. (2018) or pair-wise depth-map comparisons (Cao et al. (2018)) to add local and non-local constraints. Our approach relies on a simpler mono-scale architecture, making it efficient at run-time. Our constraints between depths, normals, and occluding contours guide learning towards good depth prediction for the whole image.

Laina et al. (2016) exploited the power of deep residual neural networks (He et al. (2016b)) and showed that using the more appropriate BerHu (Owen (2007); Zwald and Lambert-Lacroix (2012)) reconstruction loss yields better performances. However, their end results are quite smooth around occluding contours, making their method inappropriate for realistic occlusion-aware augmented reality.

Jiao et al. (2018) noticed that the depth distribution of the NYUv2 dataset is heavy-tailed. The authors therefore proposed an attention-driven loss for the network supervision, and paired the depth estimation task with semantic segmentation to improve performances on the dataset. However, while they currently achieve the best performance on the NYUv2-Depth dataset, their approach suffers from a bias towards high-depth areas such as windows, corridors or mirrors. While this translates into a significant decrease of the final error, it also produces blurry depth maps, as one can see in Fig. 3.1. By contrast, our reconstructions tend to be much sharper along the occluding boundaries as desired, and our method is much faster, making it suitable for real-time applications.

Edge- and Occlusion-Aware Depth Estimation (Wang et al. (2016)) introduced their SURGE method to improve scene reconstruction on planar and edge regions by learning to jointly predict depth and normal maps, as well as edges and planar regions. They then refine the depth prediction by solving an optimization problem using a Dense Conditional Random Field (DCRF). While their method yields appealing reconstruction results on planar regions, it still underperforms state-of-the-art methods on global metrics, and the use of DCRF makes it unsuited for real-time applications. Furthermore, SURGE (Wang et al. (2016)) is evaluated on the reconstruction quality around edges using standard depth error metrics, but not on the 2D localization of their occluding contours.

Many self-supervised methods (Godard et al. (2017); Teng et al. (2018); Yang et al. (2018b,c); Yin and Shi (2018)) have incorporated edge- or occlusion-aware geometry constraints which exist when working with stereo pairs or sequences of images as provided in the very popular KITTI depth estimation benchmark (Geiger et al. (2013)). However, although these methods can perform monocular depth estimation at test time, they require multiple calibrated views at training time. They are therefore unable to work on monocular RGB-D datasets such as NYUv2-Depth (Silberman et al. (2012)) or SUN-RGBD (Song et al. (2015)).

Wang et al. (2016) and Jiang et al. (2018) worked on occlusion-aware depth estimation to improve reconstruction for augmented reality applications. While achieving spectacular results, they however require one or multiple light-field images, which are more costly to obtain than ubiquitous RGB images.

Conscious of the lack of evaluation metrics and benchmarks for quality of edge and planes reconstruction from monocular depth estimates, Koch et al. (2018) introduced the iBims-v1 benchmark, discussed in Section 2.3.2 which introduces annotations and metrics for occluding contours and planarity of planar regions. Our evaluation method of occluding contours reconstruction quality is based on their work.

3.3 Method

As shown in Fig. 3.2, we train a network $f(\boldsymbol{I};\Theta)$ to predict, for a training color image $\boldsymbol{I}$, a depth map $\widehat{\boldsymbol{D}}$, a map of occluding contours probabilities $\widehat{\boldsymbol{C}}$, and a map $\widehat{\boldsymbol{N}}$ of surface normals. Although we focus on high quality depth-maps prediction, our occluding contours and normals map can also be used for other applications. Our approach generalizes well to various indoor datasets in terms of geometry estimation as can be seen in Fig. 3.3.

3.3.1 Training Overview

We first train f on the synthetic dataset PBRS (Zhang et al. (2017)), which provides the ground truth for the depth map $\boldsymbol{D}$, the normals map $\boldsymbol{N}$, and the binary map of object instance contours $\boldsymbol{C}$ for each training image $\boldsymbol{I}$. Since *occluding* contours are not directly provided in the PBRS dataset, we choose to use the *object instance* contours $\boldsymbol{C}$ as a proxy. We argue that on a macroscopic scale, a large proportion of occluding contours in an image are due to objects occluding one another. However, we show that we can also enable our network to learn internal occluding contours within objects even without

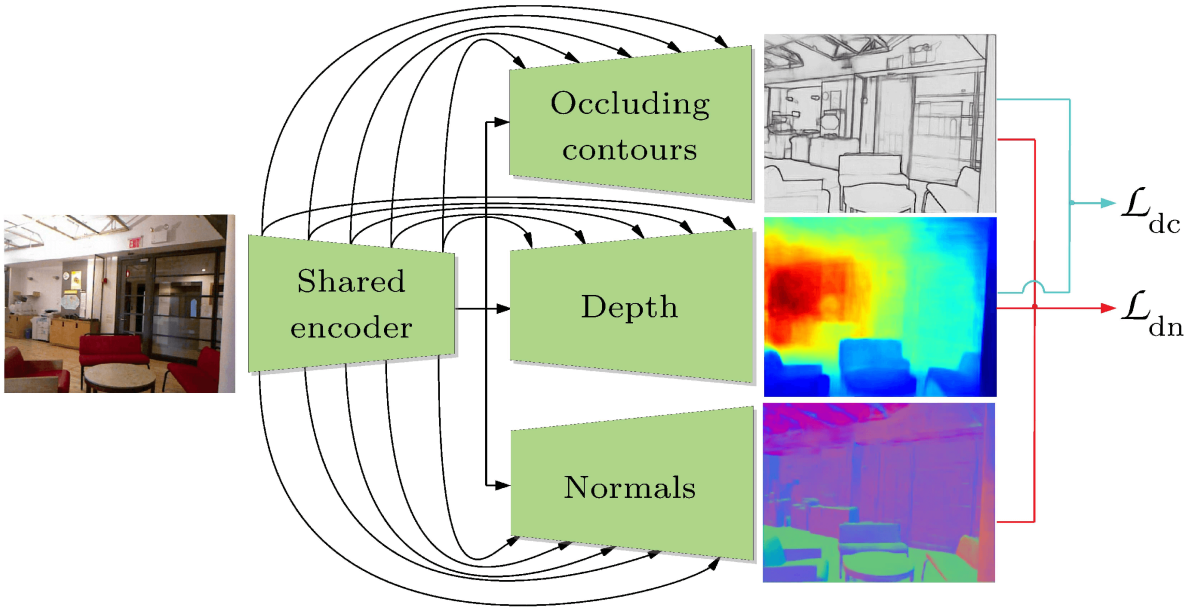


Fig. 3.2 The architecture of our "U-net"-shape (Ronneberger et al. (2015)) multi-task encoder-decoder network. We use a single ResNet50 encoder which learns an intermediate representation that is shared by all decoders. With this setting, the representation generalizes better for all tasks. We use skip connections between features of the encoder and of the decoder at corresponding scales.

"pure" occluding contours supervision. Indeed, we make use of constraints on depth map and occluding contour predictions $\hat{\boldsymbol{D}}$ and $\hat{\boldsymbol{C}}$ respectively (see Section. 3.3.4 for more details) to enforce the contour estimation task to also predict intra-object occluding boundaries.

We then finetune f on the NYUv2-Depth dataset without direct supervision on the occluding contours or normals ($\mathcal{L}_{\text{c}}$ and $\mathcal{L}_{\text{n}}$ described below): Even though Ladicky et al. (2014) and Silberman et al. (2012) produce ground truth normals map with different estimation methods operating on the Kinect-v1 depth maps, their output results are generally noisy. Occluding contours are not given in the original NYUv2-Depth dataset. Although one could automatically extract them using edge detectors (Canny (1986); Dollár and Zitnick (2015)) on depth maps, such extraction is very sensitive to the detector's parameters (see Figure 3.4). Instead, we introduce consensus terms that explicitly constrain the predicted contours, normals and depth maps together ($\mathcal{L}_{\text{dc}}$ and $\mathcal{L}_{\text{dn}}$ described below) at training time.

At test-time, we can choose to use only the depth stream of f if we are not interested in the normals nor the boundaries, making inference very fast.

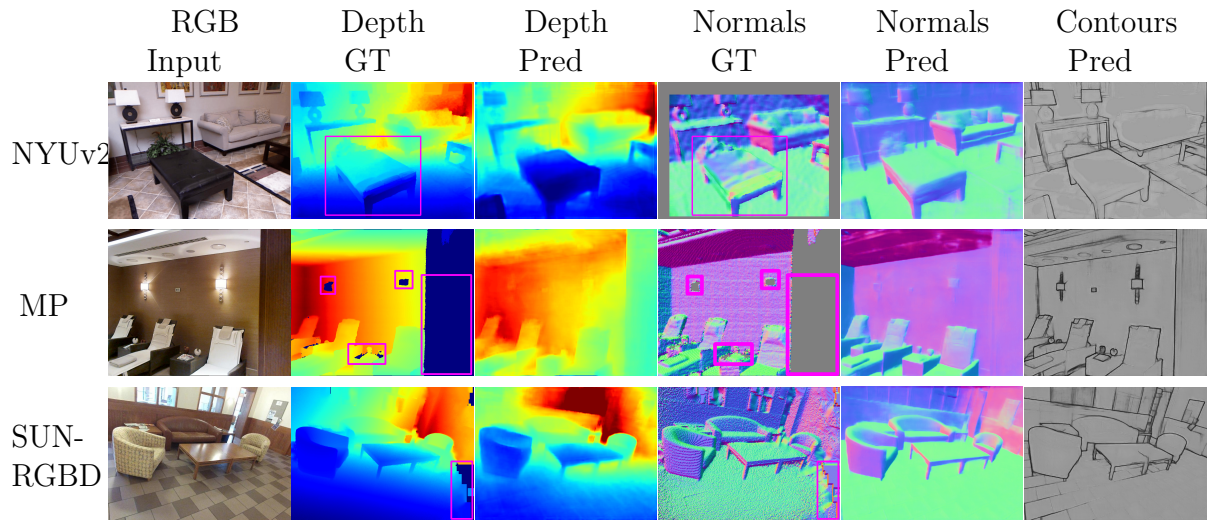


Fig. 3.3 Several predictions on single RGB images from multiple real RGB-D datasets. "MP" stands for Matterport3D, "GT" stands for ground truth and "Pred" for prediction. We highlight areas where we successfully reconstructed geometry while Kinect depth maps were inaccurate (the chair should be closer than the lamp in first image). Ground truth normals are computed using code from Silberman et al. (2012) for NYUv2 and McCormac et al. (2017) for SUN-RGBD. Normal maps are already provided in Matterport3D.

3.3.2 Loss Function

We estimate the parameters Θ of network f by minimizing the following loss function over all the training images:

$$\begin{aligned}\mathcal{L} = \lambda_{\mathrm{d}}\mathcal{L}_{\mathrm{d}}(\boldsymbol{D}, \widehat{\boldsymbol{D}}) + \lambda_{\mathrm{c}}\mathcal{L}_{\mathrm{c}}(\boldsymbol{C}, \widehat{\boldsymbol{C}}) + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}(\boldsymbol{N}, \widehat{\boldsymbol{N}}) + \\ \mathcal{L}_{\mathrm{dc}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{C}}) + \mathcal{L}_{\mathrm{dn}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{N}})\,,\end{aligned} \tag{3.1}$$

where

- $\mathcal{L}_{\mathrm{d}}$, $\mathcal{L}_{\mathrm{c}}$, and $\mathcal{L}_{\mathrm{n}}$ are supervision terms for the depth, the occluding contours, and the normals respectively. We adjust weights λ_{d}, λ_{c}, and λ_{n} during training so that we focus first on learning local geometry (normals and boundaries) then on depth. See Section 3.4.1 for more details.
- $\mathcal{L}_{\mathrm{dc}}$ and $\mathcal{L}_{\mathrm{dn}}$ introduce constraints between the predicted depth map and the predicted contours, and between the predicted depth map and the predicted normals respectively.

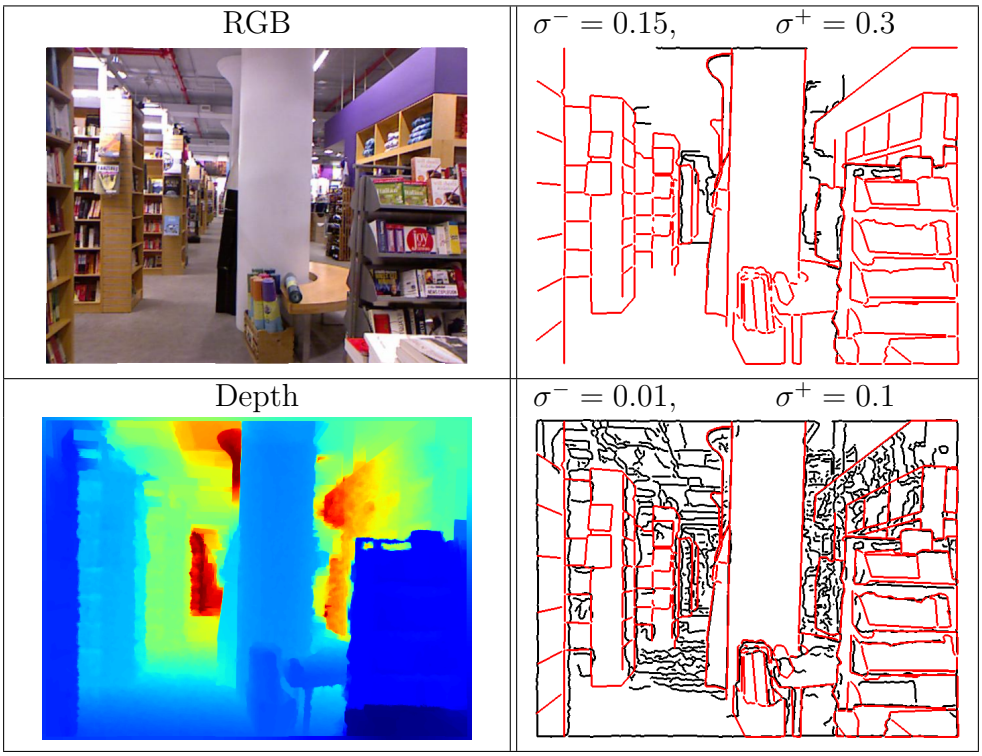


Fig. 3.4 A RGB-D sample of NYUv2-Depth for which we manually annotated occluding contours in NYUv2-OC, (in red lines). We show in black the edges detected on ground truth Kinect-v1 depth map using different Canny detector parameters (σ^- and σ^+ denote low and high threshold respectively). Highly permissive detectors often yield many spurious contours, whereas restrictive ones miss many true contours. Automatic occluding contours extraction from Kinect depth maps is therefore unreliable for extraction of ground truth occluding contours, motivating our manually annotated NYUv2-OC dataset.

We detail these losses below. All losses are computed using only valid pixel locations. The PBRS synthetic dataset provides such a mask. When finetuning on NYUv2-Depth, we mask out the white pixels on the images border.

3.3.3 Supervision Terms $\mathcal{L}_\mathbf{d}$, $\mathcal{L}_\mathbf{c}$, and $\mathcal{L}_\mathbf{n}$

The supervision terms on the predicted depth and normal maps are drawn from previous works on monocular depth prediction. For our term on occluding contours prediction, we rely on previous work for edge prediction.

Depth prediction loss $\mathcal{L}_\mathbf{d}$. As in recent works, our loss on depth prediction applies to log-distances. We use the BerHu loss function (Owen (2007); Zwald and Lambert-Lacroix

(2012)), as it was shown in Laina et al. (2016) to result in faster converging and better solutions:

$$\begin{aligned}\mathcal{L}_{\mathrm{d}}(\boldsymbol{D}, \widehat{\boldsymbol{D}}) = &\frac{1}{N}\sum_i \mathrm{BerHu}(\log(\widehat{D}_i) - \log(D_i)) \\ &+ \frac{1}{N}\sum_i \|\nabla \log(\widehat{D}_i) - \nabla \log(D_i))\|^2 \,.\end{aligned} \tag{3.2}$$

The sum is over all the N valid pixel locations. The BerHu (also known as reverse Huber) function is defined as a L_2 loss for large deviations, and a L_1 loss for small ones. As in Laina et al. (2016), we take the c parameter of the BerHu function as $c = \frac{1}{5}\max_i(|\log(\widehat{D}_i) - \log(D_i)|)$.

Occluding contours prediction loss $\mathcal{L}_{\mathbf{c}}$. We use the recent attention loss from Wang et al. (2018a), which was developed for 2d edge detection, to learn to predict the occluding contours. This attention loss helps dealing with the imbalance of edge pixels compared to non-edge pixels:

$$\mathrm{AL}(\hat{p}, p) = \begin{cases} -\alpha\beta^{(1-\hat{p})^\gamma}\log(\hat{p}) & \text{if } p = 1 \\ -(1-\alpha)\beta^{\hat{p}^\gamma}\log(1-\hat{p}) & \text{else} \end{cases} \tag{3.3}$$

where (β, γ) are hyper-parameters which we set to the authors values $(4, 0.5)$, and α is computed image per image as the proportion of contour pixels. We use this pixel-wise attention loss to define the occluding contours prediction loss:

$$\mathcal{L}_{\mathrm{c}}(\boldsymbol{C}, \widehat{\boldsymbol{C}}) = \frac{1}{N}\sum_i \mathrm{AL}(\widehat{C}_i, C_i) \,. \tag{3.4}$$

As mentioned above, this loss is disabled when finetuning on the NYUv2-Depth dataset.

Normals prediction loss $\mathcal{L}_{\mathbf{n}}$. For normals prediction, we use a common method introduced by Eigen and Fergus (2015) which is to minimize, for all valid pixels i, the angle between the predicted normals $\widehat{\boldsymbol{N}}_i$ and their ground truth counterpart $\boldsymbol{N}_i$. This angle minimization is performed by maximizing their dot-product. We therefore used the following loss:

$$\mathcal{L}_{\mathrm{n}}(\boldsymbol{N}, \widehat{\boldsymbol{N}}) = \frac{1}{N}\sum_i \left(1 - \frac{< \widehat{\boldsymbol{N}}_i, \boldsymbol{N}_i >}{\|\widehat{\boldsymbol{N}}_i\|\|\boldsymbol{N}_i\|}\right) \,. \tag{3.5}$$

This loss slightly differs from the one of Eigen and Fergus (2015) as we limit it to positive values. As mentioned earlier, this loss is disabled when finetuning on the NYUv2-Depth dataset.

3.3.4 Consensus Terms $\mathcal{L}_{\mathbf{dc}}$ and $\mathcal{L}_{\mathbf{dn}}$

Depth-contours consensus term. In order to force the network to predict sharp depth edges at occluding contours where strong depth discontinuities occur, we propose the following loss between the predicted occluding contours probability map $\widehat{\boldsymbol{C}}$ and the predicted depth map $\widehat{\boldsymbol{D}}$:

$$\begin{aligned}\mathcal{L}_{\text{dc}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{C}}) &= -\frac{1}{N}\sum_i \log(\widehat{C}_i) \cdot \|\nabla(\widehat{D}_i)\|^2 \|\Delta(\widehat{D}_i)\| \\ &+ \mu\left(\|\widehat{\boldsymbol{C}}\| - \frac{1}{N}\sum_i \log(1-\widehat{C}_i)\cdot e^{-\|\Delta(\widehat{D}_i)\|}\right).\end{aligned} \tag{3.6}$$

This encourages the network to associate pixels with large depth gradients with occluding contours: High-gradient areas will lead to a large loss unless the occluding contour probability is close to one. Godard et al. (2017) and Heise et al. (2013) also used this type of edge-aware gradient-loss, although they used it to impose consensus between photometric gradients and depth gradients. However, relying on photometric gradients can be dangerous: textured areas can exhibit strong image gradients without strong depth gradients.

Depth-normals consensus loss. Depth and normals are two highly correlated entities. Thus, to impose geometric consistency during prediction between the normal and depth predictions $\widehat{\boldsymbol{D}}$ and $\widehat{\boldsymbol{N}}$, we use the following loss:

$$\mathcal{L}_{\text{dn}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{N}}) = \frac{1}{N}\sum_i \left(1 - \frac{< \hat{\boldsymbol{u}}_i, \hat{\boldsymbol{n}}_i >}{\|\hat{\boldsymbol{u}}_i\|\|\hat{\boldsymbol{n}}_i\|}\right), \tag{3.7}$$

where $\hat{\boldsymbol{n}}_i = (\hat{n}_x^i, \hat{n}_y^i)^T$ is extracted from the 3D vector $\widehat{\boldsymbol{N}}_i = (\hat{n}_x^i, \hat{n}_y^i, \hat{n}_z^i)^T$, and $\hat{\boldsymbol{u}}_i = (\partial_x \widehat{D}_i, \partial_y \widehat{D}_i)$ is computed as the 2D gradient of the depth map estimate using finite differences. This term enforces consistency between the normals and depth predictions in a similar fashion as in Fei et al. (2018); Wang et al. (2016); Yang et al. (2018b). However, our formulation of depth-normals consensus is much simpler than those proposed in previous works as they express their constraint in 3D world coordinates, thus requiring the camera calibration matrix. Instead, we only assume that orthographic projection holds, which is a good first order assumption (Wu and Li (1988)).

Imposing this constraint during finetuning allows us to constrain normals, and depth, even when the ground truth normals $\boldsymbol{N}$ are not available (or accurate enough for our application).

3.4 Experiments

We evaluate our method and compare it to previous work using standard metrics, as well as the depth boundary edge (DBE) accuracy metric introduced by Koch et al. (2018) (see following Section 3.4.2 and Eq. (3.8) for more details). We show that our method achieves the best trade-off between global reconstruction error and DBE.

3.4.1 Implementation Details

We implement our work in Pytorch and make our pretrained weight, training and evaluation code publicly available.[1] Both training and evaluation are done on a single high-end NVIDIA GTX 1080 Ti GPU.

Datasets. We first train our network on the synthetic PBRS (Zhang et al. (2017)) dataset, using depth and normals maps annotations, along with object instance boundaries maps which we use as a proxy to occluding contours annotations. We split the PBRS dataset in training/validation/test sets using a 80%/10%/10% ratio. We then finetune our network on the NYUv2-Depth training set using only depth data. Finally, we use the NYUv2-Depth validation set for depth evaluation and our new NYUv2-OC for occluding contours accuracy evaluation.

Training. Training a multi-task network requires some caution: Since several loss terms are involved, and in particular one for each task, one should pay special attention to any suboptimal solution for one task due to 'over-learning' another. To monitor each task individually, we monitor each individual loss along with the global training loss and make sure that all of them decrease during training. When setting all loss coefficients equal to one, we noticed that the normals loss $\mathcal{L}_{normals}$ decreased faster than others. Similarly, we found that learning boundaries was much faster than learning depth. As Zhang and Funkhouser (2018), we also argue that this is because local features such as contours or local planes, *i.e.* where normals are constant, are easier to learn since they appear in almost all training examples. Training depth, however, requires the network to exploit context data such as room layout in order to regress a globally consistent depth map.

Based on those observations, we chose to learn the easier tasks first, then use them as guidance to the more complex task of depth estimation through our novel consensus loss terms of Eqs. (3.7) and (3.6). For finetuning on real data with the NYUv2 dataset, we

[1] www.github.com/MichaelRamamonjisoa/SharpNet

first disabled the consensus terms and froze the contours and normals decoders in order to first bridge the depth domain gap between PBRS and NYUv2. After convergence, we finetuned the network again with consensus terms back on, which helped enhancing predictions by ensuring consistency between geometric entities. We found that it was necessary to freeze the normals and contours decoders during finetuning to prevent their predictions $\widehat{\boldsymbol{C}}$ and $\widehat{\boldsymbol{N}}$ from degrading until being unable to play their geometry guidance role. We argue that this is due to (1) a larger synthetic-to-real domain gap for depth than for contours and normals, and (2) noisy depth ground truth with some inaccuracies along occluding contours and crease along walls. We therefore relied on the ResNet50 encoder to learn a representation which produces geometrically consistent predictions $\widehat{\boldsymbol{C}}$, $\widehat{\boldsymbol{N}}$ and $\widehat{\boldsymbol{D}}$.

3.4.2 Evaluation Method

We evaluate our method on the benchmark dataset NYUv2 Depth (Silberman et al. (2012)). The most common metrics are: Thresholded accuracies $(\delta_1, \delta_2, \delta_3)$, linear and logarithmic Root Mean Squared Error RMSE_{lin} and RMSE_{log}, Absolute Relative difference rel, and logarithmic error $\log_{10}$.

Method	Evaluated on full NYUv2-Depth							Evaluated on our NYUv2-OC			
	Accuracy ↑ ($\delta_i = 1.25^i$)			Error ↓				e^{acc}_{DBE} ↓(px) $\{\sigma^-, \sigma^+\}$			
	δ_1	δ_2	δ_3	rel	$\log_{10}$	RMSE (lin)	RMSE (log)	$\{0.1, 0.2\}$	$\{0.01, 0.1\}$	$\{0.005, 0.06\}$	$\{0.03, 0.05\}$
Eigen and Fergus (2015) (VGG)	0.766	0.949	0.988	0.195	0.068	0.660	0.217	2.895	3.065	3.199	3.203
Eigen and Fergus (2015) (AlexNet)	0.690*	0.911*	0.977*	0.250*	0.082*	0.755*	0.259*	2.840	3.029	3.202	3.242
Laina et al. (2016)	0.818	0.955	0.988	0.170	0.059	0.602	0.200	3.901	4.033	4.116	4.133
Fu et al. (2018)	0.850	0.957	0.985	0.150	0.052	0.578	0.194	3.714	3.754	4.040	4.062
Jiao et al. (2018)	**0.909**	**0.981**	**0.995**	**0.133**	**0.042**	**0.401**	**0.146**	6.389*	4.073*	4.179*	4.190*
Ours	0.888	0.979	**0.995**	0.139	0.047	0.495	0.157	**2.272**	**2.629**	**3.066**	**3.152**

Table 3.1 Our final evaluation results. Bold and underlined results indicate first and second place respectively. Asterisks indicate the last place. Numerical results might vary from the original papers, as we evaluated all methods with the same code, using only the authors depth map predictions. Results are evaluated in the center crop proposed by Eigen and Fergus (2015) and clipped depth predictions to range $[0.7m, 10m]$.

NYUv2-Depth benchmark evaluation. We have run a comparative study between our method and previous ones, summarized in Table 3.1. Since authors evaluating on the NYUv2-Depth benchmark often apply different evaluation methods, fair comparison is difficult to perform. For instance, Xu et al. (2017) and Fu et al. (2018) evaluate on crops with regions provided by Eigen and Fergus (2015). Some authors also clip resulting depth-maps to the valid depth sensor range [0.7m; 10m]. Most importantly, not all the authors make their prediction and/or evaluation code publicly available. The authors

of Jiao et al. (2018) kindly shared their predictions on the NYUv2-Depth dataset with us, and the following evaluation of their method was obtained based on the depth map predictions they provided us with. All other mentioned methods have released their predictions online.

Fair comparison is ensured by performing evaluation of each method solely using its associated depth map predictions and one single evaluation code.

Occluding contours location accuracy. To evaluate occluding contours location accuracy, we follow the work of Koch et al. (2018) as they proposed an experimental method for such evaluation. Since it is fundamental to examine whether predicted depths maps are able to represent all occluding contours as depth discontinuities in an accurate way, they analyzed occluding contours accuracy performances by detecting edges in predicted and ground truth depth maps and comparing those edges.

Since acquired depth maps in the NYUv2-Depth dataset are especially noisy around occluding boundaries, we manually annotated a subset of the dataset with occluding contours, building our NYUv2-OC dataset, which we used for evaluation. Several samples of our NYU-OC dataset are shown in Fig. 3.4 and Fig. A.4. In order to evaluate the predicted depth maps' $\boldsymbol{D}$ quality in terms of occluding contours reconstruction, binary edges $\widehat{\boldsymbol{Y}}$ are first extracted from $\widehat{\boldsymbol{D}}$ with a Canny detector.[2] They are then compared to the ground truth annotated binary edges $\boldsymbol{Y}$ from our NYU-OC dataset by measuring the a *Truncated Chamfer Distance* (TCD). Specifically, for each pixel $\widehat{Y}_i$ of $\widehat{\boldsymbol{Y}}$ we compute its euclidean distance E_i to the closest edge pixel $\widehat{Y}_j = 1$. If the distance between $\widehat{Y}_i$ and $\widehat{Y}_j$ is bigger than 10 pixels we set e_i to 0 in order to evaluate predicted edges only around the ground truth edges as seen in Fig. 3.5. This is done efficiently using *Euclidean Distance Transform* on $\boldsymbol{Y}$. The depth boundary edge (DBE) accuracy is then computed as the mean TCD over detected edges $\widehat{Y}_i = 1$:

$$\epsilon_{DBE}^{acc} = \frac{1}{\sum_i \widehat{Y}_i} \sum_i E_i \cdot \widehat{Y}_i, \tag{3.8}$$

We compare our method against state-of-the-art depth estimation methods using this metric and different Canny parameters. Evaluation results are shown in Table 3.1: We outperform all state-of-the-art methods on occluding contours accuracy, while being a competitive second best on standard depth estimation evaluation metrics.

Since the detected edges in $\widehat{\boldsymbol{Y}}$ are highly sensitive to the edge detector's parameters (see Fig.3.4), we evaluate the DBE accuracy ϵ_{DBE}^{acc} using many random combinations of

[2]Edges are extracted from depth maps with normalized dynamic range.

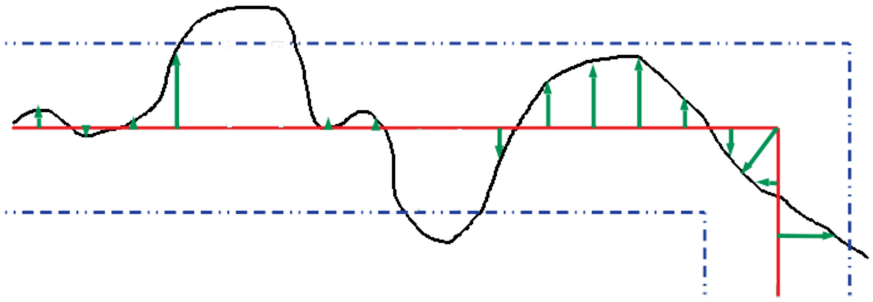


Fig. 3.5 The truncated chamfer distance is computed as the sum Euclidean distances E_i (in green) between the detected edge $\widehat{Y}_i$ (in black) and the ground truth edge Y_i (in red). The E_i above 10 pixels (above the blue dashed line) are ignored.

threshold parameters σ^+ and σ^- of the Canny edge detector. The results are shown in Fig. 3.6.

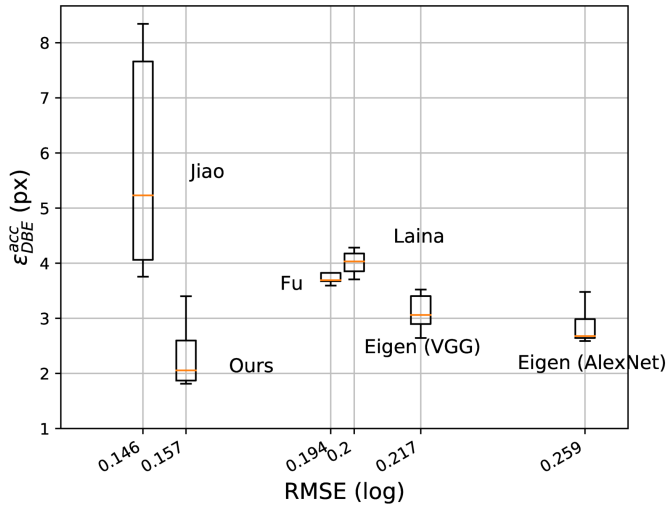


Fig. 3.6 Our method outperforms state-of-the-art in terms of trade-off between global depth reconstruction error and occluding boundary accuracy.

3.4.3 Ablation Study

To prove the impact of our geometry consensus terms, we performed an ablation study to analyze the contribution of training with synthetic and real data, as well as our novel geometry consensus terms. Evaluation of different models on our NYUv2-OC dataset are shown in Table 3.2, confirming their contribution to both improved depth reconstruction results over the whole NYUv2-Depth dataset and occluding contours accuracy.

Method	Training Dataset	$RMSE_{log}$	ϵ^{acc}_{DBE} ↓(px) $\{\sigma^-, \sigma^+\}$			
			$\{0.1, 0.2\}$	$\{0.01, 0.1\}$	$\{0.005, 0.06\}$	$\{0.03, 0.05\}$
w/o consensus	PBRS	0.304*	2.321	2.751*	3.298*	3.380*
w/ consensus	PBRS	0.262	**2.046**	**2.332**	**2.574**	**2.645**
w/o consensus	PBRS + NYUv2	<u>0.163</u>	2.600*	2.638	3.127	3.182
w/ consensus	PBRS + NYUv2	**0.157**	<u>2.272</u>	<u>2.629</u>	<u>3.066</u>	<u>3.152</u>

Table 3.2 Our added geometry consensus terms brings a significant performance boost by guiding the depth towards learning accurate occluding contours and it also helps keeping a good trade-off between occluding contours accuracy and depth reconstruction during the necessary fine-tuning on real RGB-D data. $RMSE_{log}$ is computed over the full NYUv2-Depth dataset. Notations of Table. 3.1 are used here.

3.5 Conclusion

In this chapter, we show that our SharpNet method is able to achieve competitive depth reconstruction from a single RGB image with particular attention to occluding contours thanks to geometry consensus terms introduced during multi-task training. Our high-quality depth estimation which yields high accuracy occluding contours reconstruction allows for realistic integration of virtual objects in real-time augmented reality as we achieve 150 fps inference speed. We show the superiority of our SharpNet over state-of-the-art by introducing a first version of our new NYUv2-OC occluding contours dataset, which we plan to extend in future work. As by-products of our approach, high-quality normals and contours predictions can also be a useful representation for other computer vision tasks. More results can be found in Appendix A.

While this first step towards sharper depth edges achieves higher quality results than state-of-the-art, we build upon its success by exploring another class of neural networks, Spatial Transformer Networks (Jaderberg et al. (2015)), which is used to develop our Displacement Fields, which we discuss in the next chapter.

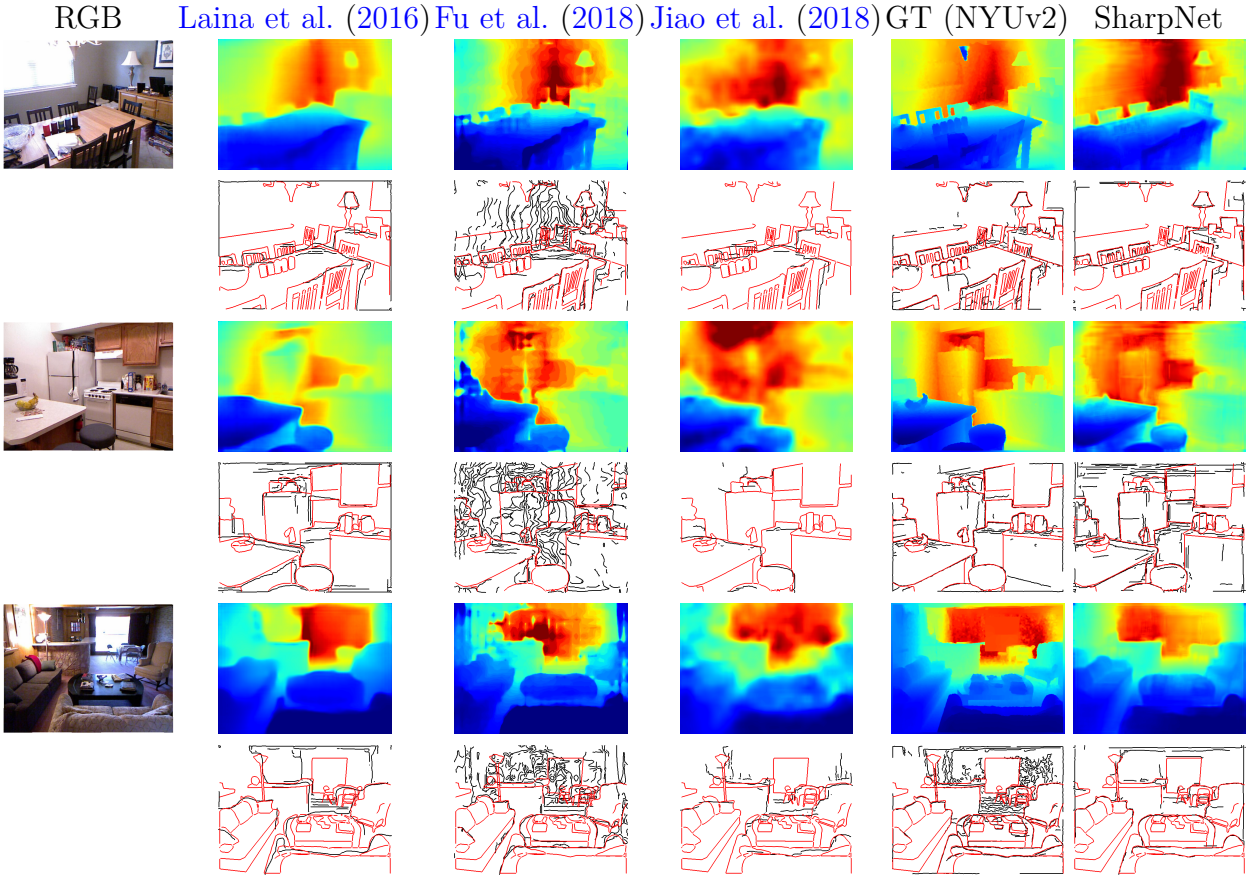


Fig. 3.7 Several examples of images from our NYUv2-OC dataset and their associated depth map estimate for different methods. The second row for each image shows the in black the detected edges on those estimates using a Canny edge detector (in black) with $\sigma^- = 0.03$ and $\sigma^+ = 0.05$, overlaid on our manually annotated ground truth in red. Our SharpNet method not only creates sharper occluding contours, leading to less spurious and erroneous contours than with Fu et al. (2018) the Kinect-v1 depth-map; it also leads to much better located edges than other methods.

Chapter 4

Estimating Sharper Depth Maps with Displacement Fields

The work described in this chapter is based on the following publication:
Predicting Sharp and Accurate Occlusion Boundaries in Monocular Depth Estimation Using Displacement Fields, Michaël Ramamonjisoa, Yuming Du and Vincent Lepetit published at CVPR 2020.

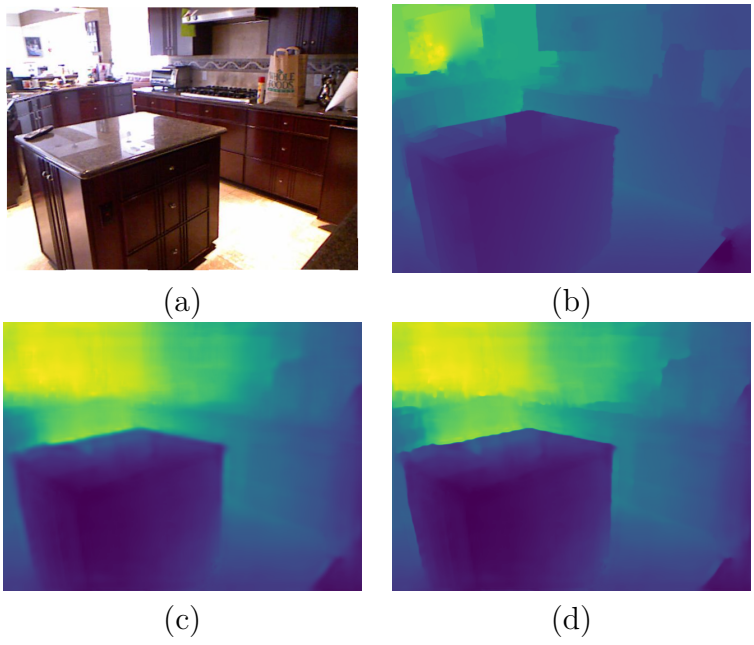


Fig. 4.1 (a) Input image, (b) Ground truth depth from NYUv2-Depth, (c) Predicted depth using SharpNet (Ramamonjisoa and Lepetit (2019)), (d) Refined depth using our pixel displacement method.

Abstract

Current methods for depth map prediction from monocular images tend to predict smooth, poorly localized contours for the occlusion boundaries in the input image. This is unfortunate as occlusion boundaries are important cues to recognize objects, and as we show, may lead to a way to discover new objects from scene reconstruction. To improve predicted depth maps, recent methods rely on various forms of filtering or predict an additive residual depth map to refine a first estimate. We instead learn to predict, given a depth map predicted by some reconstruction method, a 2D displacement field able to re-sample pixels around the occlusion boundaries into sharper reconstructions. Our method can be applied to the output of any depth estimation method and is fully differentiable, enabling end-to-end training. For evaluation, we manually annotated the occlusion boundaries in all the images in the test split of popular NYUv2-Depth dataset. We show that our approach improves the localization of occlusion boundaries for all state-of-the-art monocular depth estimation methods that we could evaluate (Eigen and Fergus (2015); Fu et al. (2018); Jiao et al. (2018); Laina et al. (2016)), without degrading the depth accuracy for the rest of the images.

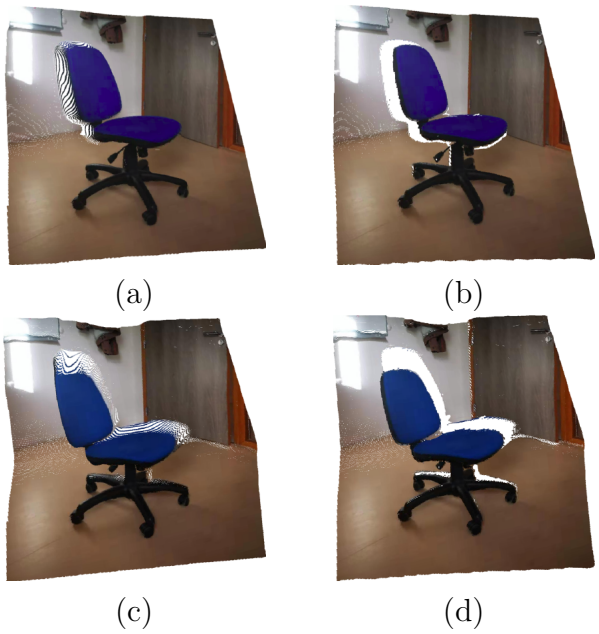


Fig. 4.2 Application of our depth map refinement to 3D object extraction. (a-b) and (c-d) are two point cloud views of our extracted object. The left column shows point clouds extracted from the initially predicted depth map (by Ramamonjisoa and Lepetit (2019)), while the right one shows the result after using our depth refinement method. Our method suppresses long tails around object boundaries, as we achieve sharper occlusion boundaries.

4.1 Introduction

As discussed in Chapter 3 obtaining sharp and accurate occlusion boundaries in depth maps predicted from a single image is an important and challenging task.

As demonstrated in Figure 4.1, despite the recent advances presented in Chapter 3, the occlusion boundaries in the predicted depth maps still remain poorly reconstructed. However we believe that this direction is particularly important: Depth prediction generalizes well to unseen objects and even to unseen object categories, and being able to reconstruct well the occlusion boundaries could be a promising line of research for unsupervised object discovery. In Figure 4.2 we show how our work allows for better object extraction from background.

In this chapter, we introduce a simple method to overcome smooth occlusion boundaries. Our method improves their sharpness as well as their localization in the images. It relies on a differentiable module that takes an initial depth map provided by some depth prediction method, and re-sample it to obtain more accurate occlusion boundaries. Optionally, it can also take the color image as additional input for guidance information

and obtain even better contour localization. This is done by training a deep network to predict a 2D displacement field, applied to the initial depth map. This contrasts with previous methods that try to improve depth maps by predicting residual offsets for depth values (Jeon and Lee (2018); Zhang et al. (2019)). We show that predicting displacements instead of such a residual helps reaching sharper occluding boundaries. We believe this is because our module enlarges the family of functions that can be represented by a deep network. As our experiments show, this solution is complementary with all previous solutions as it systematically improves the localization and reconstruction of the occlusion boundaries.

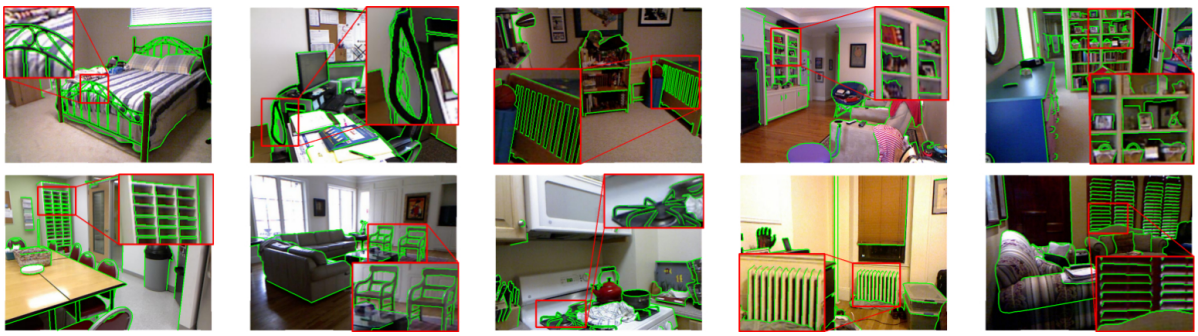


Fig. 4.3 Samples of our NYUv2-OC++ dataset, which extends NYUv2-OC from Ramamonjisoa and Lepetit (2019). The selected highlighted regions in red rectangles emphasize the high-quality and fine-grained annotations.

In order to improve the evaluation of occlusion boundary reconstruction performance of existing MDE methods and of our proposed method, we extended the NYUv2-OC dataset introduced in Chapter 3, and manually annotated the occlusion boundaries in all the images of the NYUv2 test set. Some annotations are shown in Fig. 4.3 and in the supplementary material. Based on the evaluation process and metrics introduced in Koch et al. (2018) and discussed in Chapter 3, we show that our method quantitatively improves the performance of all state-of-the-art MDE methods in terms of localization accuracy while maintaining or improving the global performance in depth reconstruction on two benchmark datasets.

In the rest of the chapter, we first discuss related work. We then present our approach to sharpen occlusion boundaries in depth maps. Finally, we describe our experiments and results to prove how our method improves state-of-the-art MDE methods.

4.2 Related Work

Occlusion boundaries are a notorious and important challenge in dense reconstruction from images, for multiple reasons. For example, it is well known that in the case of stereo reconstruction, some pixels in the neighborhood of occluding boundaries are hidden in one image but visible in the other image, making the task of pixel matching difficult. In this context, numerous solutions have already been proposed to alleviate this problem (Fua (1991); Geiger et al. (1995); Intille and Bobick (1994); Kanade and Okutomi (1994)). We focus here on recent techniques used in monocular depth estimation to improve the reconstruction of occlusion boundaries.

In Chapter 3, we discussed previous works that focuses on improving depth estimation with edge-aware constraints. In this section, we cover related work that aims to refine initial depth maps predictions, as well as datasets that were previously released to evaluate image contours detection methods.

Depth Refinement Methods In this section we discuss two main approaches that can be used to refine depth maps predictions: We first discuss the formerly popular Conditional Random Fields (CRFs), then classical filtering methods and their newest versions.

Several previous work used CRFs post-processing potential to refine depth maps predictions. These works typically define pixel-wise and pair-wise loss terms between pixels and their neighbors using an intermediate predicted guidance signal such as geometric features (Wang et al. (2016)) or reliability maps (Heo et al. (2018)). An initially predicted depth map is then refined by performing inference with the CRF, sometimes iteratively or using cascades of CRFs (Xu et al. (2017, 2018a)). While most of these methods help improving the initial depth predictions and yield qualitatively more appealing results, those methods still under-perform state-of-the-art non-CRF MDE methods while being more computationally expensive.

Another option for depth refinement is to use image enhancement methods. Even though these methods do not necessarily explicitly target occlusion boundaries, they can be potential alternative solutions to ours and we compare with them in Section 4.4.5.

Bilateral filtering is a popular and currently state-of-the-art method for image enhancement, in particular as a denoising method preserving image contours. Although historically, it was limited for post-processing due to its computational complexity, recent work have successfully made bilateral filters reasonably efficient and fully differentiable (Li et al. (2016); Wu et al. (2018a)). These recent methods have been successful when applied in downsampling-upsampling schemes, but have not been used yet in the context of MDE.

Guided filters (He et al. (2013)) have been proposed as an alternative simpler version of the bilateral filter. We show in our experiments that both guided and bilateral filters sharpen occlusion boundaries thanks to their usage of the image for guidance. They however sometime produce false depth gradients artifacts. The bilateral solver (Barron and Poole (2016)) formulates the bilateral filtering problem as a regularized least-squares optimization problem, allowing fully differentiable and much faster computation. However, we show in our experiments that our end-to-end trainable method compares favorably against this method, both in speed and accuracy.

Datasets with Image Contours Several datasets of image contours or occlusion boundaries already exist. Popular datasets for edge detection training and evaluation were focused on perceptual (Arbelaez et al. (2011); Martin et al. (2004, 2001)) or object instance (Everingham et al. (2010)) boundaries. However, those datasets often lack annotation of the occlusion relationship between two regions separated by the boundaries. Other datasets (Hoiem et al. (2011, 2007); Ren et al. (2006); Wang and Yuille (2016)) annotated occlusion relationship between objects, however they do not contain ground truth depth.

The NYUv2-Depth dataset (Silberman et al. (2012)) is a popular MDE benchmark which provides such depth ground truth. Several methods for instance boundary detection have benefited from this depth information (Deng et al. (2018); Dollár and Zitnick (2013); Gupta et al. (2013); Gupta et al. (2014); Ren and Bo (2012)) to improve their performances on object instance boundaries detection.

The above cited datasets all lack object self-occlusion boundaries and are sometimes inaccurately annotated. Our NYUv2-OC++ dataset provides manual annotations for the occlusion boundaries on top of NYUv2-Depth for all of its 654 test images. As discussed in Chapter 3, even though it is a tedious task, manual annotation is much more reliable than automated annotation that could be obtained from depth maps. Figure 4.3 illustrates the extensive and accurate coverage of the occlusion boundaries of our annotations. This dataset enables simultaneous evaluation of depth estimation methods and occlusion boundary reconstruction as the 100 images iBims dataset (Koch et al. (2018)), but is larger and has been widely used for MDE evaluation.

4.3 Method

In this section, we introduce our occlusion boundary refinement method as follows. Firstly, we present our hypothesis on the typical structure of predicted depth maps

around occlusion boundaries and derive a model to transform this structure into the expected one. We then prove our hypothesis using an hand-crafted method. Based on this model, we propose an end-to-end trainable module which can resample pixels of an input depth image to restore its sharp occlusion boundaries.

4.3.1 Prior Hypothesis

Occlusion boundaries correspond to regions in the image where depth exhibits large and sharp variations, while the other regions tend to vary much smoother. Due to the small proportion of such sharp regions, neural networks tend to predict over-smoothed depths in the vicinity of occlusion boundaries.

We show in this work that sharp and accurately located boundaries can be recovered by resampling pixels in the depth map. This resampling can be formalized as:

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p} + \boldsymbol{\delta p}(\boldsymbol{p})) \,, \tag{4.1}$$

where D is a depth map, $\boldsymbol{p}$ denotes an image location in domain Ω, and $\boldsymbol{\delta p}(\boldsymbol{p})$ a 2D displacement that depends on $\boldsymbol{p}$. This formulation allows the depth values on the two sides of occlusion boundaries to be "stitched" together and replace the over-smoothed depth values. While this method shares some insights with Deformable CNNs J. Dai et al. (2017) where the convolutional kernels are deformed to adapt to different shapes, our method is fundamentally different as we displace depth values using a predicted 2D vector for each image location.

Another option to improve depth values would be to predict an additive residual depth, which can be formalized as, for comparison:

$$\forall \boldsymbol{p} \in \Omega, \quad D(\boldsymbol{p}) \leftarrow D(\boldsymbol{p}) + \Delta D(\boldsymbol{p}) \,. \tag{4.2}$$

We argue that updating the predicted depth $\widehat{\boldsymbol{D}}$ using predicted pixel shifts to recover sharp occlusion boundaries in depth images works better than predicting the residual depth. We validate this claim with the experiments presented below on toy problems, and on real predicted depth maps in Section 4.4.

4.3.2 Testing the Optimal Displacements

To first validate our assumption that a displacement field $\boldsymbol{\delta p}(\boldsymbol{p})$ can improve the reconstructions of occlusion boundaries, we estimate the optimal displacements using ground

truth depth for several predicted depth maps as:

$$\forall \boldsymbol{p} \in \Omega, \boldsymbol{\delta p}^* = \underset{\boldsymbol{\delta p}:\ \boldsymbol{p}+\boldsymbol{\delta p}\ \in \mathcal{N}(\boldsymbol{p})}{\arg\min} (D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p}+\boldsymbol{\delta p}))^2 \,. \tag{4.3}$$

In words, solving this problem is equivalent to finding for each pixel the optimal displacement $\boldsymbol{\delta p}^*$ that reconstructs the ground truth depth map D from a predicted depth map $\widehat{D}$. We solve Eq. (4.3) by performing for all pixels $\boldsymbol{p}$ an exhaustive search of $\boldsymbol{\delta p}$ within a neighborhood $\mathcal{N}(\boldsymbol{p})$ of size 50×50. Qualitative results are shown in Fig. 4.4. The depth map obtained by applying this optimal displacement field is clearly much better.

In practice, we will have to predict the displacement field to apply this idea. This is done by training a deep network, which is detailed in the next subsection. We then check on a toy problem that this yields better performance than predicting residual depth with a deep network of similar capacity.

4.3.3 Method Overview

Based on our model, we propose to learn the displacements of pixels in predicted depth images using CNNs. Our approach is illustrated in Fig. 4.6: Given a predicted depth image $\widehat{D}$, our network predicts a displacement field $\boldsymbol{\delta p}$ to resample the image locations in depth map $\widehat{D}$ according to Eq. (4.1). This approach can be implemented as a Spatial Transformer Network (Jaderberg et al. (2015)).

Image guidance can be helpful to improve the occlusion boundary precision in refined depth maps and can also help to discover edges that were not visible in the initial predicted depth map $\widehat{D}$. However, it should be noted that our network can still work even without image guidance.

4.3.4 Training Our Model on Toy Problems

In order to verify that displacement fields $\boldsymbol{\delta p}$ presented in Section 4.3.1 can be learned, we first define a toy problem in 1D.

In this toy problem, as shown in Fig. 4.5, we model the signals D to be recovered as piecewise continuous functions, generated as sequences of basic functions such as step, affine and quadratic functions. These samples exhibit strong discontinuities at junctions and smooth variations everywhere else, which is a property similar to real depth maps. We then convolve the D signals with random-size (blurring) Gaussian kernels to obtain smooth versions $\widehat{D}$. This gives us a training set $\mathcal{T}$ of $(\widehat{D}, D)$ pairs.

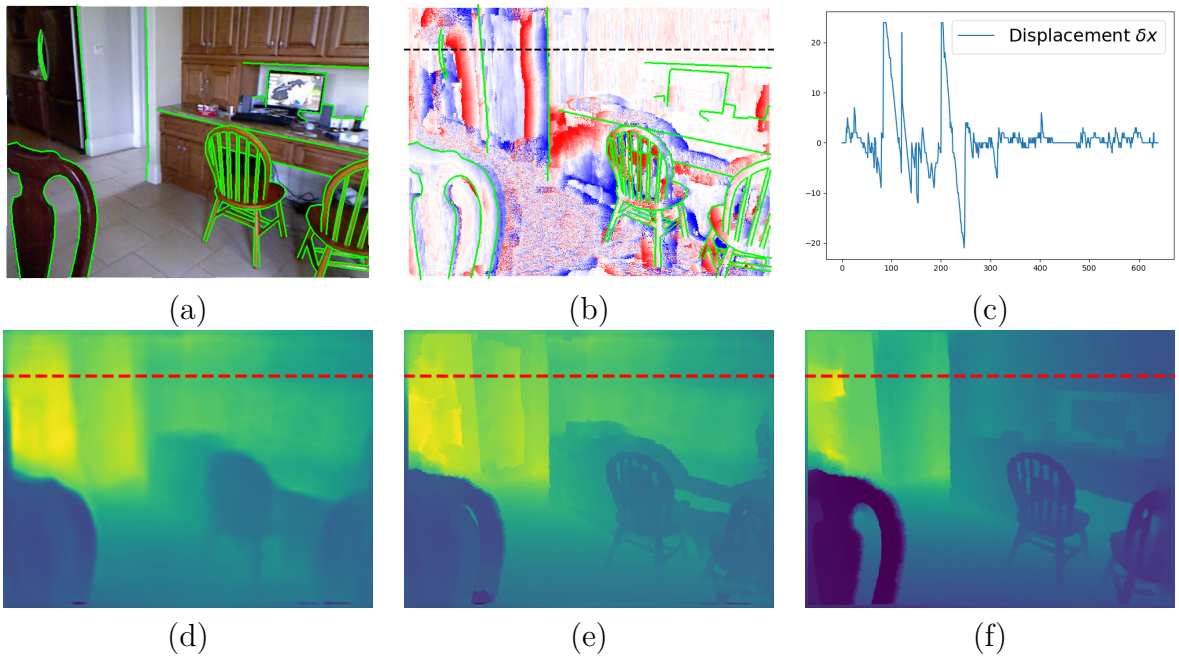


Fig. 4.4 Refinement results using the gold standard method described in Section 4.3.2 to recover the optimal displacement field (best seen in color). (a) is the input RGB image with superimposed NYUv2-OC++ annotation in green and (d) its associated Ground Truth depth. (e) is the prediction using Laina et al. (2016) with pixel displacements $\boldsymbol{\delta p}$ from Eq. (4.3) and (f) the refined prediction. (b) is the horizontal component of the displacement field $\boldsymbol{\delta p}^*$ obtained by Eq. (4.3). Red and blue color indicate positive and negative values respectively. (c) is the horizontal component δx of displacement field $\boldsymbol{\delta p}^*$ along the dashed red line drawn in (b,c,d,e).

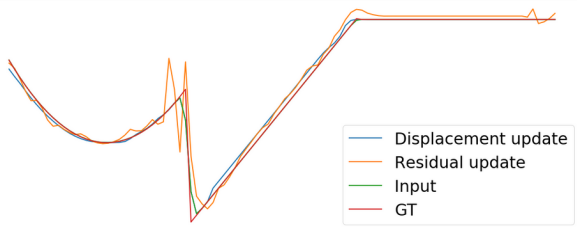


Fig. 4.5 Comparison between displacement and residual update learning. Both residual and displacement learning can predict sharper edges, however residual updates often produce artifacts along the edges while displacements update does not.

We use $\mathcal{T}$ to train a network $f(.;\Theta_f)$ of parameters to predict a displacement field:

$$\min_{\Theta_f} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}\left(\boldsymbol{p} + f(\widehat{D};\Theta_f)(\boldsymbol{p})\right)\right) . \tag{4.4}$$

and a network $g(.;\Theta_g)$ of parameters to predict a residual depth map:

$$\min_{\Theta_g} \sum_{(\widehat{D},D)\in\mathcal{T}} \sum_{\boldsymbol{p}} L\left(D(\boldsymbol{p}) - \widehat{D}(\boldsymbol{p}) + g(\widehat{D};\Theta_g)(\boldsymbol{p})\right) , \tag{4.5}$$

where $L(.)$ is some loss. In our experiments, we evaluate the l_1, l_2, and Huber losses.

As shown in Fig. 4.5, we found that predicting a residual update produces severe artifacts around edges such as overshooting effects. We argue that these issues arise because the residual CNN g is also trained on regions where the values of $\widehat{D}$ and D are different even away from edges, thus encouraging network g to also correct these regions. By contrast, our method does not create such overshooting effects as it does not alter the local range of values around edges.

It is worth noticing that even when we allow $\widehat{D}$ and D to have slightly different values in non-edge areas -which simulates residual error between predicted and ground truth depth-, our method still converges to a good solution, compared to the residual CNN.

We extend our approach validation from 1D to 2D, where the 1D signal is replaced by 2D images with different polygons of different values. We apply the same operation to smooth the images and then use our network to recover the original sharp images from the smooth ones. We observed similar results in 2D: The residual CNN always generates artifacts. Some results of our 2D toy problem can be found in supplementary material.

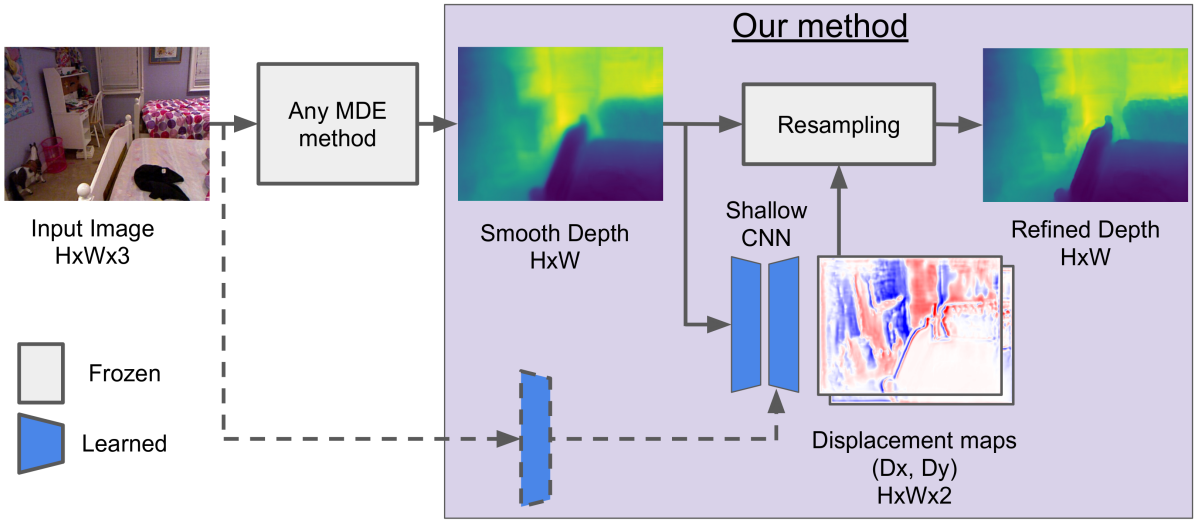


Fig. 4.6 Our proposed pipeline for depth edge sharpening. The dashed lines define the optional guidance with RGB features for our shallow network.

4.3.5 Learning to Sharpen Depth Predictions

To learn to produce sharper depth predictions using displacement fields, we first trained our method in a similar fashion to the toy problem described in Section 4.3.4. While this already improves the quality of occlusion boundaries of all depth map predictions, we show that we can further improve quantitative results by training our method using predictions of an MDE algorithm on the NYUv2-Depth dataset as input and the corresponding ground truth depth as target output. We argue that this way, the network learns to correct more complex distortions of the ground truth than Gaussian blurring. In practice, we used the predictions of SharpNet (Ramamonjisoa and Lepetit (2019)) to demonstrate this, as it is state-of-the-art on occlusion boundary sharpness. We show that this not only improves the quality of depth maps predicted by SharpNet but all other available MDE algorithms. Training our network using the predictions of other algorithms on the NYUv2 would further improve their result, however not all of them provide their code or their predictions on the official *training set*. Finally, our method is fully differentiable. While we do not do it in this paper, it is also possible to train an MDE jointly with our method, which should yield even better results.

(a) (b) (c) (d) (e) (f)

Fig. 4.7 Refinement results using our method (best seen in color). From left to right: (a) input RGB image with NYUv2-OC++ annotation in green, (b) SharpNet (Ramamonjisoa and Lepetit (2019)) depth prediction, (c) Refined prediction, (d) Ground truth depth, (e) Horizontal and (f) Vertical components of the displacement field. Displacement fields are clipped between ±15 pixels. Although SharpNet is used as an example here because it is currently state-of-the-art on occlusion boundary accuracy, similar results can be observed when refining predictions from other methods.

4.4 Experiments

In this section, we first detail our implementation, describe the metrics we use to evaluate the reconstruction of the occlusion boundaries and the accuracy of the depth prediction, and then present the results of the evaluations of our method on the outputs from different MDE methods.

4.4.1 Implementation Details

We implemented our network using the Pytorch framework (Paszke et al. (2019)). Our network is a light-weight encoder-decoder network with skip connections, which has one encoder for depth, an optional one for guidance with the RGB image, and a shared decoder. Details on each component can be found in the supplementary material. We trained our network on the output of a MDE method (Ramamonjisoa and Lepetit (2019)), using Adam optimization with an initial learning rate of 5e-4 and weight decay of 1e-6, and the *poly* learning rate policy (Zhao et al. (2017)), during 32k iterations on NYUv2 (Silberman et al. (2012)). This dataset contains 1449 pairs RGB and depth images, split into 795 samples for training and 654 for testing. Batch size was set to 1. The input images were resized with scales $[0.75, 1, 1, 5, 2]$ and then cropped and padded to 320×320. We tried to learn on the *raw* depth maps from the NYUv2 dataset, without

success. This is most likely due to the fact that missing data occurs mostly around the occlusion boundaries. Dense depth maps are therefore needed which is why we use Levin et al. (2004) to inpaint the missing data in the raw depth maps.

4.4.2 Evaluation metrics

4.4.2.1 Evaluation of monocular depth prediction

As in previous work (Eigen and Fergus (2015); Eigen et al. (2014); Laina et al. (2016)), we evaluate the monocular depth predictions using the following metrics: Root Mean Squared linear Error (R_{lin}), mean absolute relative error (rel), mean $\log_{10}$ error ($\log_{10}$), Root Mean Squared log Error (R_{log}), and the accuracy under threshold ($\sigma_i < 1.25^i$, $i = 1, 2, 3$).

4.4.2.2 Evaluation of occlusion boundary accuracy

Following the work of Koch et al. (2018), we evaluate the accuracy of occlusion boundaries (OB) using the proposed depth boundary errors, which evaluate the accuracy ϵ_a and completion ϵ_c of predicted occlusion boundaries. The boundaries are first extracted using a Canny edge detector (Canny (1986)) with predefined thresholds on a normalized predicted depth image. As illustrated in Fig. 4.8, ϵ_a is taken as the average Chamfer distance in pixels (Fan et al. (2017)) from the predicted boundaries to the ground truth boundaries; ϵ_{com} is taken as the average Chamfer distance from ground truth boundaries to the predicted boundaries.

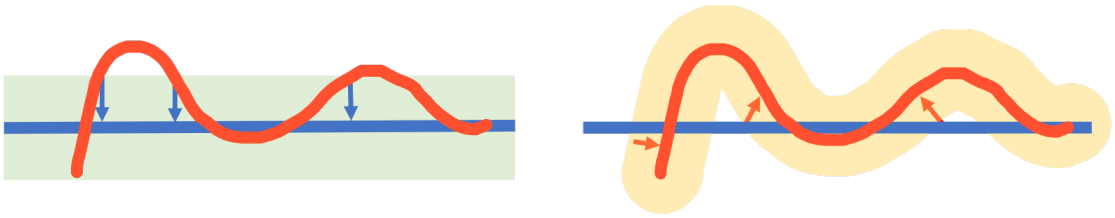
4.4.3 Evaluation on the NYUv2 Dataset

We evaluate our method by refining the predictions of different state-of-the-art methods (Eigen et al. (2014); Fu et al. (2018); Jiao et al. (2018); Laina et al. (2016); Ramamonjisoa and Lepetit (2019); Yin et al. (2019)). Our network is trained using the 795 labeled NYUv2 depth images of training dataset with corresponding RGB images as guidance.

To enable a fair comparison, we evaluate only pixels inside the crop defined in Eigen et al. (2014) for all methods. Table 4.1 shows the evaluation of refined predictions of different methods using our network. With the help of our proposed network, the occlusion boundary accuracy of all methods can be largely improved, without degrading the global depth estimation accuracy. We also show qualitative results of our refinement method in Fig. B.4.

Method	Refine	Depth error (↓)				Depth accuracy (↑)			OBs (↓)	
		rel	$\log_{10}$	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Eigen et al. (2014)	-	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
	✓	0.232	0.094	0.758	0.263	0.615	0.889	0.971	2.168	8.173
Laina et al. (2016)	-	0.142	0.059	0.510	0.181	0.818	0.955	0.988	4.702	8.982
	✓	0.140	0.059	0.509	0.180	0.819	0.956	0.989	2.372	7.041
Fu et al. (2018)	-	0.131	0.053	0.493	0.174	0.848	0.956	0.984	3.872	8.117
	✓	0.136	0.054	0.502	0.178	0.844	0.954	0.983	3.001	7.242
Ramamonjisoa and Lepetit (2019)	-	0.116	0.053	0.448	0.163	0.853	0.970	0.993	3.041	8.692
	✓	0.117	0.054	0.457	0.165	0.848	0.970	0.993	1.838	6.730
Jiao et al. (2018)	-	0.093	0.043	0.356	0.134	0.908	**0.981**	**0.995**	8.730	9.864
	✓	**0.092**	**0.042**	**0.352**	**0.132**	**0.910**	**0.981**	**0.995**	2.410	8.230
Yin et al. (2019)	-	0.112	0.047	0.417	0.144	0.880	0.975	0.994	1.854	7.188
	✓	0.112	0.047	0.419	0.144	0.879	0.975	0.994	**1.762**	**6.307**

Table 4.1 Evaluation of our method on the output of several state-of-the-art methods on NYUv2. Our method significantly improves the occlusion boundaries metrics ϵ_a and ϵ_c without degrading the other metrics related to the overall depth accuracy. These results were computed using available depth maps predictions (apart from Jiao *et al.* Jiao et al. (2018) who sent us their predictions) within the image region proposed in Eigen et al. (2014). (↓: Lower is better; ↑: Higher is better).



(a) (b)

Fig. 4.8 Occlusion boundary evaluation metrics based on the Chamfer distance, as introduced in Koch et al. (2018). The blue lines represent the ground truth boundaries, the red curve the predicted boundary. Only the boundaries in the green area are taken into account during the evaluation of accuracy (a), and only the yellow area are taken into account during the evaluation of completeness (b). (a) The accuracy is evaluated from the distances of points on the predicted boundaries to the ground truth boundaries. (b) The completeness is evaluated from the distances of points on the ground truth boundaries to the predicted boundaries.

4.4.4 Evaluation on the iBims Dataset

We applied our method trained on NYUv2 dataset to refine the predictions from various methods (Eigen and Fergus (2015); Eigen et al. (2014); Laina et al. (2016); Li et al. (2017); Liu et al. (2018, 2015); Ramamonjisoa and Lepetit (2019)) on the iBims dataset(Koch et al. (2018)). Table 4.2 shows that our network significantly improves the accuracy and completeness metrics for the occluding boundaries of all predictions on this dataset as well.

4.4.5 Comparison with Other Methods

To demonstrate the efficiency of our proposed method, we compare our method with existing filtering methods (Barron and Poole (2016); He et al. (2013); Tomasi and Manduchi (1998); Wu et al. (2018a)). We use the prediction of Eigen et al. (2014) as input, and compare the accuracy of depth estimation and occlusion boundaries of each method. Note that for the filters with hyper-parameters, we tested each filter with a series of hyper-parameters and select the best refined results. For the Fast Bilateral Solver (FBS) (Barron and Poole (2016)) and the Deep Guided Filter (GF) (Wu et al. (2018a)), we use their default settings from the official implementation. We keep the same network with and without Deep GF, and train both times with the same learning rate and data augmentation.

Method	Refine	Depth error (↓)			Depth accuracy (↑)			OB (↓)	
		rel	$\log_{10}$	R_{lin}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Eigen et al. (2014)	-	0.32	0.17	1.55	0.36	0.65	0.84	9.97	9.99
	✓	0.32	0.17	1.54	0.37	0.66	0.85	4.83	8.78
Eigen and Fergus (2015) (AlexNet)	-	0.30	0.15	1.38	0.40	0.73	0.88	4.66	8.68
	✓	0.30	0.15	1.37	0.41	0.73	0.88	4.10	7.91
Eigen and Fergus (2015) (VGG)	-	0.25	0.13	1.26	0.47	0.78	0.93	4.05	8.01
	✓	0.25	0.13	1.25	0.48	0.78	0.93	3.95	7.57
Laina et al. (2016)	-	0.26	0.13	1.20	0.50	0.78	0.91	6.19	9.17
	✓	0.25	0.13	1.18	0.51	0.79	0.91	3.32	7.15
Liu et al. (2015)	-	0.30	0.13	1.26	0.48	0.78	0.91	2.42	7.11
	✓	0.30	0.13	1.26	0.48	0.77	0.91	2.36	7.00
Li et al. (2017)	-	**0.22**	**0.11**	1.09	0.58	**0.85**	**0.94**	3.90	8.17
	✓	**0.22**	**0.11**	1.10	0.58	0.84	**0.94**	3.43	7.19
Liu et al. (2018)	-	0.29	0.17	1.45	0.41	0.70	0.86	4.84	8.86
	✓	0.29	0.17	1.47	0.40	0.69	0.86	2.78	7.65
Ramamonjisoa and Lepetit (2019)	-	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	3.69	7.82
	✓	0.27	**0.11**	**1.08**	**0.59**	0.83	0.93	**2.13**	**6.33**

Table 4.2 Evaluation of our method on the output of several state-of-the-art methods on iBims.

As shown in Table 4.3, our method achieves the best accuracy for occlusion boundaries. Finally, we compare our method against the additive residual prediction method discussed in 4.3.1. We keep the same U-Net architecture, but replace the displacement operation with an addition as described in Eq. (4.2), and show that we obtain better results. We argue that the performance of the deep guided filter and additive residual are lower due to generated artifacts which are discussed in Section 4.3.4. In Table 4.4, we also compare our network's computational efficiency against reference depth estimation and refinement methods.

4.4.6 Influence of the Loss Function and the Guidance Image

In this section, we report several ablation studies to analyze the favorable factors of our network in terms of performances. Fig. 4.6 shows the architecture of our baseline network. All the following networks are trained using the same setting detailed in Section 4.4.1 and trained on the official 795 RGB-D images of the NYUv2 training set. To evaluate the effectiveness of our method, we choose the predictions of Eigen et al. (2014) as input, but note that the conclusion is still valid with other MDE methods. Please see supplementary material for further results.

Method	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline (Eigen et al. (2014))	0.234	0.095	0.766	0.265	0.610	0.886	*0.971*	9.926	9.993
Bilateral Filter (Tomasi and Manduchi (1998))	0.236	0.095	0.765	0.265	0.611	0.887	*0.971*	9.313	9.940
Guided Filter (He et al. (2013))	0.237	0.095	0.767	0.265	0.610	0.885	*0.971*	6.106	9.617
FBS (Barron and Poole (2016))	0.236	0.095	0.765	0.264	0.611	0.887	*0.971*	5.428	9.454
Deep GF (Wu et al. (2018a))	0.306	0.116	0.917	0.362	0.508	0.823	0.948	4.318	9.597
Residual	0.286	0.132	0.928	0.752	0.508	0.807	0.931	5.757	9.785
Our Method	**0.232**	**0.094**	**0.757**	**0.263**	**0.615**	**0.889**	*0.971*	**2.302**	**8.347**

Table 4.3 Comparison with existing methods for image enhancement, adapted to the depth map prediction problems. Our method performs the best for this problem over all the different metrics.

Method	SharpNet (Ramamonjisoa and Lepetit (2019))	VNL (Yin et al. (2019))	Deep GF (Wu et al. (2018a))	Ours
FPS - GPU	83.2 ± 6.0	32.2± 2.1	70.5 ± 7.5	**100.0** ± 7.3
FPS - CPU	2.6 ± 0.0	*	4.0 ± 0.1	**5.3** ± 0.15

Table 4.4 Speed comparison with other reference methods implemented using Pytorch. Those numbers were computed using a single GTX Titan X and Intel Core i7-5820K CPU. using 320x320 inputs. Runtime statistics are computed over 200 runs.

4.4.6.1 Loss Functions for Depth Prediction

In Table 4.5, we show the influence of different loss functions. We apply the Pytorch official implementation of l_1, l_2, and the Huber loss. The Disparity loss supervises the network with the reciprocal of depth, the target depth y_{target} is defined as $y_{target} = M/y_{original}$, where M here represents the maximum of depth in the scene. As shown in Table 4.5, our network trained with l_1 loss achieves the best accuracy for the occlusion boundaries.

4.4.6.2 Guidance image

We explore the influence of different types of guidance image. The features of guidance images are extracted using an encoder with the same architecture as the depth encoder, except that Leaky ReLU (Maas et al. (2013)) activations are all replaced by standard ReLU (Glorot et al. (2011a)). We perform feature fusion of guidance and depth features using skip connections from the guidance and depth decoder respectively at similar scales.

	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
Method	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline Eigen et al. (2014)	0.234	0.095	0.766	0.265	0.610	0.886	0.971	9.926	9.993
l_1	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**
l_2	*0.232*	*0.094*	**0.757**	*0.263*	*0.615*	*0.889*	0.971	2.302	8.347
Huber	*0.232*	0.095	0.758	*0.263*	*0.615*	*0.889*	**0.972**	2.225	8.282
Disparity	0.234	0.095	0.761	0.264	0.613	0.888	0.971	2.312	8.353

Table 4.5 Evaluation of different loss functions for learning the displacement field. The l_1 norm yields the best results.

Table 4.6 shows the influence of different choices for the guidance image. The edge images are created by accumulating the detected edges using a series of Canny detector with different thresholds. As shown in Table 4.6, using the original RGB image as guidance achieves the highest accuracy, while using the image converted to grayscale achieves the lowest accuracy, as information is lost during the conversion. Using the Canny edge detector can help to alleviate this problem, as the network achieves better results when switching from gray image to binary edge maps.

	Depth error (↓)				Depth accuracy (↑)			OB (↓)	
Method	rel	log10	R_{lin}	R_{log}	σ_1	σ_2	σ_3	ϵ_a	ϵ_c
Baseline (Eigen et al. (2014))	0.234	0.095	0.760	0.265	0.612	0.886	0.971	9.936	9.997
No guidance	0.236	0.096	0.771	0.268	0.608	0.883	0.969	6.039	9.832
Gray	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.659	8.681
Binary Edges	*0.232*	*0.094*	*0.757*	*0.263*	*0.615*	*0.889*	*0.972*	2.466	8.483
RGB	*0.232*	*0.094*	0.758	*0.263*	*0.615*	*0.889*	0.971	**2.168**	**8.173**

Table 4.6 Evaluation of different ways of using the input image for guidance. Simply using the original color image works best.

4.5 Conclusion

We showed that by predicting a displacement field to resample depth maps, we can significantly improve the reconstruction accuracy and the localization of occlusion boundaries of any existing method for monocular depth prediction. To evaluate our method, we also introduce a new dataset of precisely labeled occlusion boundaries. Beyond evaluation of occlusion boundary reconstruction, this dataset should be valuable for future methods to learn to detect more precisely occlusion boundaries. Following Chapter 3, this chapter introduces a new state-of-the-art on sharpness of depth edges obtained by MDE methods,

at a minimal computational cost. In the following chapter, we discuss a new method to improve efficiency of MDE architectures by exploiting the structure of depth maps.

Chapter 5

Improving the Efficiency of Monocular Depth Estimation using Wavelets Transforms

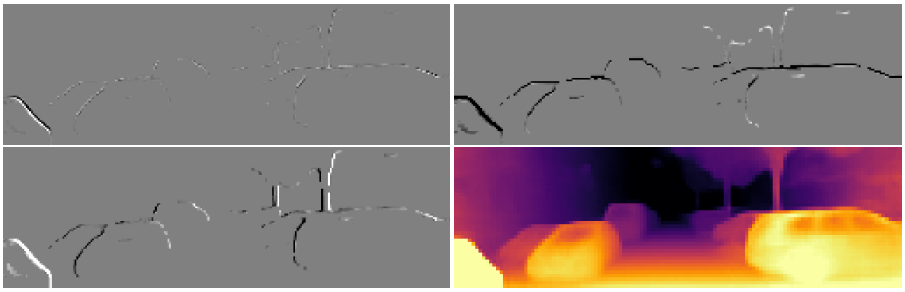
The work described in this chapter is based on the following publication:
Single Image Depth Prediction with Wavelet Decomposition, Michaël Ramamonjisoa, Michael Firman, Jamie Watson, Vincent Lepetit and Daniyar Turmukhambetov, published at CVPR 2021.

Abstract

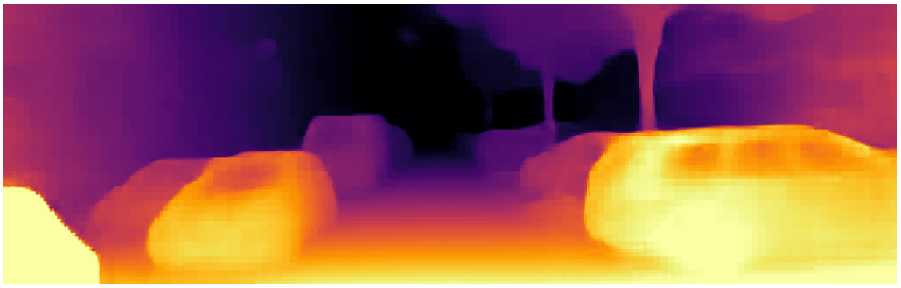
We present a novel method for predicting accurate depths from monocular images with high efficiency. This optimal efficiency is achieved by exploiting wavelet decomposition, which is integrated in a fully differentiable encoder-decoder architecture. We demonstrate that we can reconstruct high-fidelity depth maps by predicting sparse wavelet coefficients.

In contrast with previous works, we show that wavelet coefficients can be learned without direct supervision on coefficients. Instead we supervise only the final depth image that is reconstructed through the inverse wavelet transform. We additionally show that wavelet coefficients can be learned in fully self-supervised scenarios, without access to ground-truth depth. Finally, we apply our method to different state-of-the-art monocular depth estimation models, in each case giving similar or better results compared to the original model, while requiring less than half the multiply-adds in the decoder network.

(a) Input color image – (320×1024)



(b) Sparse estimation using wavelets. Our network up-samples and refines a 1/16-resolution depth map (bottom-right), by estimating wavelet coefficients only in sparse regions.



(c) Reconstruction of the output depth map using the inverse wavelet transform.

Fig. 5.1 We can represent depth maps more efficiently with wavelets. Here the network takes image (a) as input and outputs a low resolution depth map, together with sparse wavelet coefficients (b). We can reconstruct a high-resolution depth map (c) using the inverse wavelet transform. In our model we *predict* multi-scale wavelet coefficients with an image-to-image network, and we exploit sparseness of the output to save computation.

5.1 Introduction

As seen in Section 1.2, single-image depth estimation methods are useful in many real-time applications, for example robotics, autonomous driving and augmented reality. These areas are typically resource-constrained, so efficiency at prediction time is important.

Similarly to the works presented in Chapters 3 and 4, neural networks which estimate depth from a single image overwhelmingly use U-Net architectures, with skip connections between encoder and decoder layers (Ronneberger et al. (2015)). Most work on single-image depth prediction has focused on improved depth accuracy, without focusing on efficiency. Those that have cared about efficiency have typically borrowed tricks from the "efficient network" world (Howard et al. (2017); Sandler et al. (2018)) to make faster depth estimation, with the network using standard convolutions all the way through (Poggi et al. (2018a); Wofk et al. (2019)). All these approaches still use standard neural network components: convolutions, additions, summations and multiplications.

Inspired by sparse representations that can be achieved with wavelet decomposition, we propose an alternative network representation for more efficient depth estimation, using *wavelet decomposition*. We call this system **WaveletMonodepth**. We make the observation that depth images of the man-made world are typically made up of many piece-wise flat regions, with a few 'jumps' in depth between the flat regions. This structure lends itself well to wavelets. A low-frequency component can represent the overall scene structure, while the 'jumps' can be well captured in high-frequency components. Crucially, the high-frequency components are sparse, which means computation can be focused only in certain areas. This has the effect of saving run-time computation, while still enabling high-quality depths to be estimated.

To the best of our knowledge, we are the first to train a single-image depth estimation network that reconstructs depth by predicting wavelet coefficients. Furthermore, we show that our models can be trained with self-supervised loss on the final depth signal, in contrast to other methods that directly supervise predicted wavelet coefficients.

We evaluate on NYU and KITTI datasets, where we train supervised and self-supervised, respectively. We show that our approach allows us to effectively trade off depth accuracy against runtime computation.

5.2 Related Work

While most previous works in monocular depth estimation, discussed in Section 2.3 achieve higher scores with equivalently trained architectures, some works aim for improved depth

accuracy *at the expense of efficiency*. Our wavelet-based method aims to improve depth estimation efficiency, without sacrificing accuracy. In this section, we thus first discuss previous works in computer vision that used wavelets, then present related methods to improve efficiency of neural networks in general, then especially the few works that focused on efficient monocular depth estimation.

5.2.1 Wavelets in Computer Vision

Wavelet decomposition is an extensively used technique in signal processing, image processing and computer vision. The discrete wavelet transform (DWT) allows a representation of a discrete signal which is more redundant and hence compressible. A notable example is compression of images with JPEG2000 format (Taubman and Marcellin (2013); Unser and Blu (2003)). Furthermore, wavelet decomposition is also a frequency transform, and can be used for denoising (Donoho (1995); Donoho and Johnstone (1994); Kang et al. (2018)). Wavelet transforms have also recently been combined with Deep Learning to restore images affected by Moiré color artifacts, which occur when RGB sensors are unable to resolve high-frequency details (Liu et al. (2020b); Luo et al. (2020b)). Li et al. (2020a) show that by substituting pooling operations in neural networks with discrete wavelet transforms it is possible to filter out high-frequency components of the input image during prediction and thus improve noise-robustness in image classification tasks. Super-resolution methods (Deng et al. (2019); Guo et al. (2017); Huang et al. (2017b)) learn to estimate the high-frequency wavelet coefficients of an input low-resolution image to generate high-frequency image through inverse wavelet transform.

Closer to our work, Yang et al. (2020) use wavelets in a stereo matching network but require supervision of wavelet coefficients while we do not. Similarly, Luo et al. (2020a) replaced the down-sampling and up-sampling operations of UNet-like architectures with DWTs and inverse DWT respectively, and replaced standard skip-connection with high-frequency coefficient skip-connections. However, they do not directly predict wavelet coefficients of depth and as such are unable to exploit the sparse representation of wavelets for efficiency. In contrast with both these works, we focus on efficient depth prediction *from a single image*.

5.2.2 Efficient Neural Networks

Convolutional Neural Networks (CNNs) (LeCun et al. (1995)) have revolutionized the field of computer vision as CNN based methods tend to outperform every other competing methods on regression or classification tasks, if they are provided enough training data.

However, the best performing neural networks contain a large number of parameters and require a large number of floating point operations (FLOPs) at runtime, making deployment to lightweight platforms problematic. Many architectures have been developed to improve the accuracy/speed tradeoff in deep nets. For example, depth-wise separable convolutions (Howard et al. (2017)), inverted residual layers (Sandler et al. (2018)), and pointwise group convolutions (Zhang et al. (2018)). An alternative approach though is to train a network before cutting down some of its unnecessary computations.

Channel pruning. One line of research is *network pruning* (He et al. (2017b); Liu et al. (2017); Yu et al. (2019)), which consists of removing some of the redundant filters in a trained neural network. While this helps reducing the network memory footprint as well as the number of FLOPs necessary for inference, sparsity is typically enforced through regularization terms He et al. (2017b); Wen et al. (2016) to compress the network without losing performance. Using such regularisation, however, often requires careful tuning to achieve the desired result (Ye et al. (2018)). In contrast, our wavelet-based method intrinsically provides sparsity in outputs and intermediate activations, and the wavelet predictions coincide with *edges* in the depth map, knowledge of which has direct applications *e.g.* in augmented reality (Holynski and Kopf (2018); Ramamonjisoa and Lepetit (2019)).

While most works focus on classification, channel pruning has also been successfully applied to depth estimation in the aforementioned FastDepth (Wofk et al. (2019)), which uses NetAdapt (Yang et al. (2018a)) to perform channel pruning.

Sparse inference. Another recent work considers spatially sparse inference in image-to-image translation tasks. PointRend (Kirillov et al. (2020)) treats semantic segmentation as a rendering process, where a high-resolution estimate is obtained from a low-resolution one through a cascade of upsampling and sparse refinement operations. The location of these sparse *rendering* operations is chosen based on an uncertainty measure of the classification method. However, while they demonstrate the efficiency and applicability of their method to classification tasks, their method cannot directly be applied to regression tasks because of the requirement to evaluate an uncertainty heuristic for all pixel locations. In contrast, our method can directly be applied to regression tasks, as *rendering* locations are directly predicted by our model as non-zero-valued high-frequency wavelet coefficients.

Efficient depth estimation. Only a small number of works have been developed for light-weight MDE. FastDepth (Wofk et al. (2019)) exploits separable convolutions, channel pruning and efficient encoder to compress U-Net based neural network architectures.

Poggi et al. (2018a) introduce PyDNet, which uses an image pyramid to enable high receptive field with a small number of parameters. Huynh et al. (2022) explored Neural Architecture Search (NAS) to build a light-weight MDE system. Several works also used knowledge distillation to enable a small depth estimation network to learn some of the knowledge from a larger network (Liu et al. (2020c); Spek et al. (2018)).

In contrast to these works, our contribution is to change the internal representation of depth within the network itself. We note that our contributions could be used in conjunction with the above efficient architectures or distillation schemes.

5.3 Method

In this section, we first introduce the basics of 2D wavelet transforms. We chose Haar wavelets(Haar (1910)) due to their simplicity and provided efficiency. Next, we describe how to use the cascade nature of wavelet representations to build our efficient depth estimation architecture, which we call **WaveletMonodepth**. Finally, we discuss the computational benefits of sparse representations.

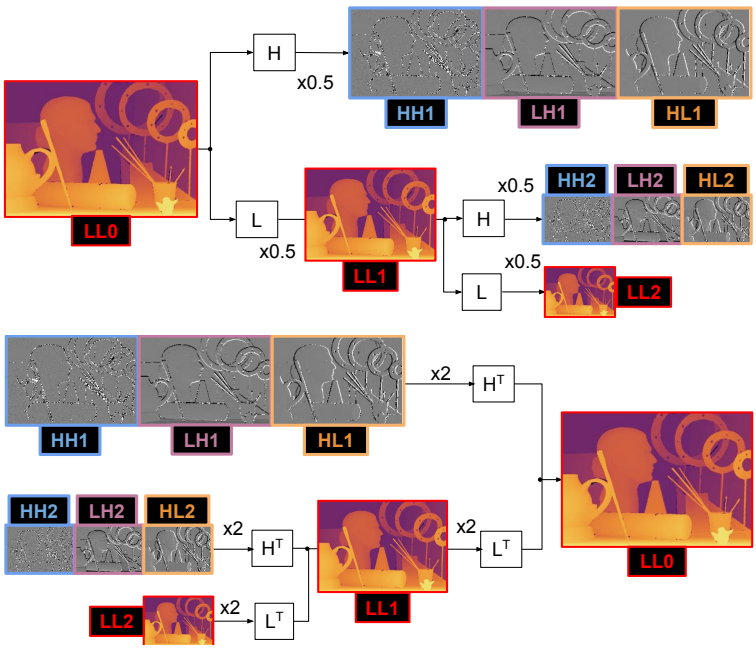
5.3.1 Haar Wavelet Transform

The Haar wavelet basis is the simplest basis of functions for wavelet decomposition. A discrete wavelet transform (DWT) with Haar wavelets decomposes a 2D image into four coefficient maps: a low-frequency (L) component LL and three high-frequency (H) components LH, HL, HH at half the resolution of the input image. For the remainder of the paper, we refer to the coefficient maps as the output of the **DWT**. The **DWT** is an invertible operation, where its inverse, **IDWT**, converts four coefficient maps into a 2D signal at twice the resolution of the coefficient maps.

The multi-scale and multi-frequency wavelet representation is build by recursively applying **DWT** to the low-frequency coefficient map LL, starting from the input image—see Figure 5.2(a). Similarly, the multi-scale representation can be recursively inverted to reconstruct a full resolution image (Figure 5.2(b)). This synthesis operation is the building block of our depth reconstruction method.

5.3.2 WaveletMonoDepth

Our method, which we call WaveletMonoDepth, is summarized in Figure 5.3. It builds on a recursive use of **IDWT** operation applied to predicted coefficient maps. Thus, we reconstruct a depth map at the input scale by first predicting a coarse estimate at the



(a) Forward transform (DWT). This decomposes the high resolution input (left) into multi-scale low resolution outputs.

(b) Inverse transform (IDWT). This reconstructs the original input from estimated low resolution inputs.

Fig. 5.2 Illustration of a two-level wavelet representation of a depth image. The input image LL_0 is passed through a two-level wavelet decomposition (a), to produce a low frequency depth map together with associated wavelets for high frequency detail. The inverse wavelet transform (b) can reconstruct the original image from the wavelet decomposition.

bottleneck scale of a UNet-like architecture (Ronneberger et al. (2015)), and iteratively upscale and refine this estimate by predicting high-frequency coefficient maps.

In our network architecture, the coarse depth estimate LL_3 is estimated at 1/16 of the input scale. This depth map is then progressively upscaled and refined using Algorithm 1. A forward pass of our model generates a collection of 5 depth maps LL_s for scales [1/16, 1/8, 1/4, 1/2, 1]. We choose to supervise only the four last scales as in Monodepth2 (Godard et al. (2019)). It is worth noting that the coefficient maps are predicted at scales [1/16, 1/8, 1/4, 1/2], thus removing the need for full-resolution computation.

5.3.3 Sparse Computations in Decoder

For piecewise flat depth maps, high-frequency coefficient maps have a small number of non-zero values; these are located around depth edges. Hence, for full-resolution depth reconstruction, only some pixel locations need to predict non-zero coefficient map values at each scale. At any scale, we assume that these pixel locations with non-zero values can be determined from high-frequency coefficient maps estimated at the previous scale defined by a mask M described in `GetSparseMask` of Algorithm 1.

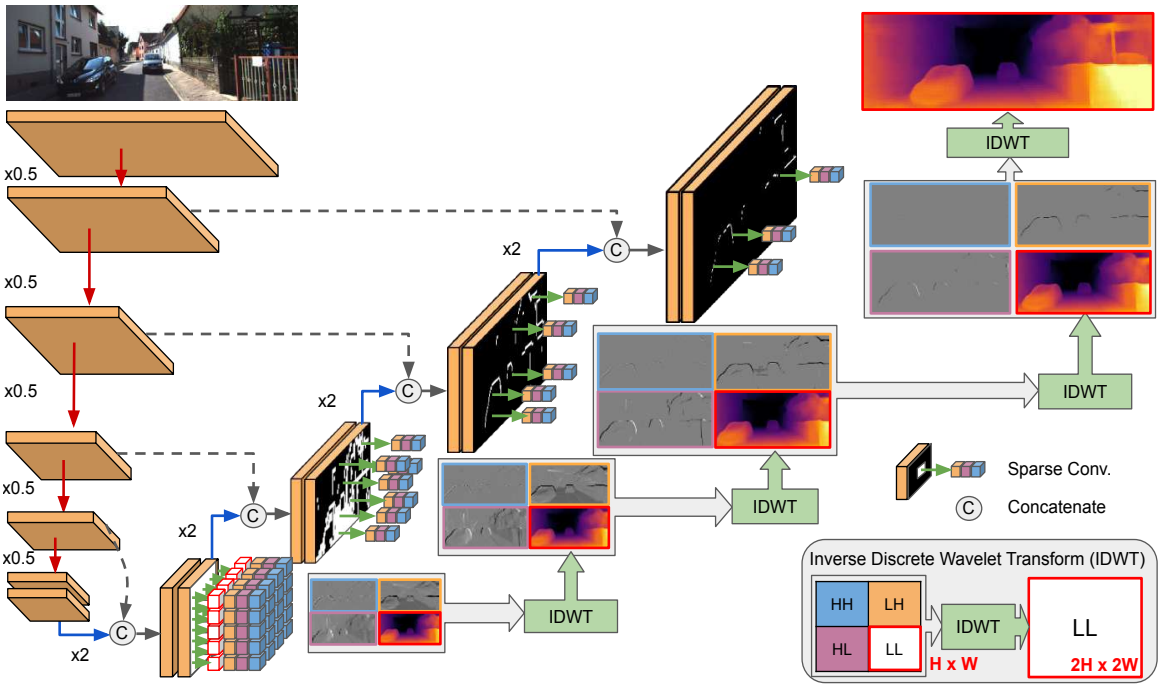


Fig. 5.3 Our method WaveletMonoDepth predicts depth from a single image using wavelets. At each stage in our decoder, we predict sparse wavelet coefficients $\{\mathsf{LH}, \mathsf{HL}, \mathsf{HH}\}$. These capture the high-frequency details of the depth map, *e.g.* occlusion boundaries. These are combined with the low-frequency depth map LL, taken from the previous level in the decoder, and passed through an inverse discrete wavelet transform (IDWT). This generates a new depth map at twice the resolution of LL. This process is continued through the decoder until the original input image resolution is reached. Because the wavelet coefficients are *sparse*, we can save computations; we need only to evaluate each decoder layer at the *non-zero* wavelet locations in the previous level. See Algorithm 1 for more details.

Algorithm 1: Computing depth with wavelets

Result: Depth map at input scale
Input : Pyramid of feature maps [F_4, F_3, F_2, F_1];
Current scale $s = 3$;
$\mathsf{LL}_3 \leftarrow$ `DensePredict` (F_4);
Threshold η ;
Sparse computation mask M = Initialize with 1;
for ($s = 3;\ s >= 0;\ s = s - 1$) {
  $\leftarrow \mathsf{LH}_s, \mathsf{HL}_s, \mathsf{HH}_s \leftarrow$ `SparsePredict` (F_{s+1}; M);
  $\mathsf{LL}_{s-1} \leftarrow$ **IDWT**(LL_s, [LH_s, HL_s, HH_s]);
  $\eta_s \leftarrow \eta \cdot (\max(\mathsf{LL}_{s-1}) - \min(\mathsf{LL}_{s-1}))$;
  $M \leftarrow$ `GetSparseMask` (LH_s, HL_s, HH_s, η_s);
}
procedure `SparsePredict`(*F, M*)
  Input : Feature map F, Sparse mask M
  for (all p s.t. $M[p] == 1$) {
    H[p] = SparseConv3x3(F[p]);
  }
  return H;
procedure `DensePredict`(*F*)
  Input : Feature map F
  Initialize M with ones;
  return `SparsePredict`(*F, M*);
procedure `GetSparseMask`(*H, η*)
  Input : High frequency coefficient maps H
  $M = \max(|\mathsf{LH}|, |\mathsf{HL}|, |\mathsf{HH}|) > \eta$;
  $M = \mathtt{upsample}_{\times 2}(M)$;
  return M;

The sparsity level achieved by using mask M is

$$\psi = \frac{\sum_{r,c=1,1}^{H,W} M_{r,c}}{HW}, \tag{5.1}$$

which allows us to remove redundant computation in the decoder layer. Indeed, for a typical $K \times K$ convolution (with a bias term) on a feature tensor of size $H \times W$ that has C_{in} input channels and C_{out} output channels, the number of multiply-add operations is

$$\text{MAC}_{\text{dense}} = HW(C_{\text{in}}K^2 + 1)C_{\text{out}}. \tag{5.2}$$

With the sparsity level ψ, it would be

$$\text{MAC}_{\text{sparse}} = \psi HW(C_{\text{in}}K^2 + 1)C_{\text{out}}. \tag{5.3}$$

Note that our sparsification strategy aims to reduce FLOPs by decreasing the number of pixel locations at which we need to compute an output. This approach is orthogonal and complements other approaches such as channel pruning, which instead reduces C_{in} and C_{out}, or separable convolutions. We refer to supplementary material for further details on these.

Considering a quite conservative threshold $\eta = 0.05$ used on high-frequency coefficient maps, the sparse decoder computation is about $3\times$ lower in FLOPs compared to standard convolutions at all pixel locations for an image of size 320×1024.

5.3.4 Self-supervised Training

Our self-supervised losses are as described in Godard et al. (2019), which we briefly describe here for completeness. See supplemental material for further details. Given a stereo pair of images (I_L, I_R), we train our network to predict a depth map D_L, pixel-aligned with the left image. We also assume access to the camera intrinsics K, and the relative camera transformation between the images in the stereo pair $T_{R\to L}$. We use the network's current estimate of depth to synthesise an image $I_{R\to L}$, computed as

$$I_{R\to L} = I_R\Big\langle \text{proj}(D_L, T_{R\to L}, K)\Big\rangle, \tag{5.4}$$

where proj() are the 2D pixel coordinates obtained by projecting the depths D_L into image I_R, and $\big\langle\big\rangle$ is the sampling operator. We follow standard practice in training with a photometric reconstruction error pe, so our loss becomes $L_p = pe(I_L, I_{R\to L})$. Following Chen et al. (2019c); Godard et al. (2019) etc., we set pe to a weighted sum of SSIM and L_1 losses.

We also include the depth smoothness loss from Monodepth2 (Godard et al. (2019)).

For our experiments which train on monocular and stereo sequences ('MS'), we combine reprojection errors from the three different source images: one frame forward in time, one frame back in time, and the corresponding stereo pair. In this case, we create synthesized images from the monocular sequence using relative poses estimated from a pose network, as described in Monodepth2. In this setting, we use a per-pixel minimum reprojection loss, again following Monodepth2.

5.4 Experiments

Our validation experiments explore the task of training a CNN to predict depth from a single color image, using wavelets as an intermediate representation. Depending on the experiment, we compare against known leading baselines that supplement, and pre- and post-process the stereo pairs used for supervision, and the output depth maps.

5.4.1 Implementation Details

Datasets We conduct experiments on the KITTI and NYUv2 depth datasets. KITTI (Geiger et al. (2012)) consists of 22,600 calibrated stereo video pairs captured by a car driving around a city in Germany. Models are evaluated using the Eigen split (Eigen and Fergus (2015)) using corresponding LiDAR point clouds; see *e.g.* Godard et al. (2017) for details. NYUv2 (Silberman et al. (2012)) consists of RGBD frames captured with a Kinect sensor. There are 120K raw frames collected by scanning various indoor scenes. As in DenseDepth (Alhashim and Wonka (2018)), we use a 50K samples subset of the full dataset where depth is inpainted using Levin et al. (2004) inpainting method. The NYUv2 evaluation is run on the 654 test frames introduced by Eigen et al. (2014).

Models To demonstrate the efficiency of our method, we choose two models for experiments on the NYUv2 and KITTI datasets. Both models are implemented using Pytorch and use a compatible implementation of **IDWT** (Cotter (2019)).

For KITTI, we choose the weakly-supervised Depth Hints (Watson et al. (2019)) method, which adds Semi Global Matching (Hirschmuller (2005, 2007)) supervision to the self-supervised Monodepth2 (Godard et al. (2019)), without requiring Lidar depth supervision. At each scale s of the Monodepth2 decoder there is a layer which outputs a one-channel disparity. We replace this layer at each scale with a 3-channel output layer to predict $\{\mathsf{LH}_s, \mathsf{HL}_s, \mathsf{HH}_s\}$. While our baseline consumes decoder feature maps at scales [1/16, 1/8, 1/4, 1/2, 1], we only need to keep the four scales [1/16, 1/8, 1/4, 1/2], as the **IDWT** outputs disparity at 2× resolution. Both our model and baseline are trained with an Adam optimizer using a learning rate of 10^{-4}, with batch size 12 for 20 epochs. Unless otherwise specified, our experiments are done with Resnet50-based model trained with depth hints loss at 320×1024 resolution.

For NYUv2, we implement a UNet-like baseline similar to DenseDepth Alhashim and Wonka (2018), and detail its architecture in supplementary material. Similar to our KITTI experiments, we discard the last layer of the decoder as it is not needed, and add one extra layer at each scale to predict the wavelet coefficients. Both our

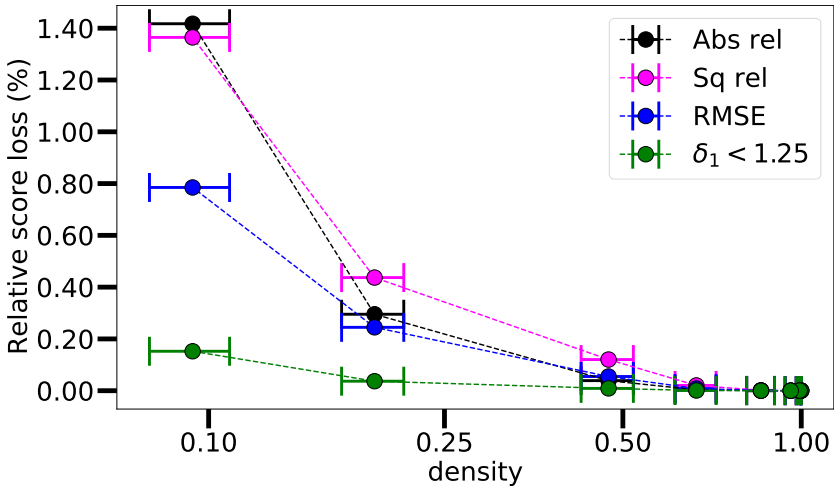


Fig. 5.4 Analysis of performance loss vs density on KITTI. Using our wavelet representation, we can drop up to 90% of the wavelet coefficients while suffering a maximum relative performance loss of less than 1.4%.

model and baseline are trained using an Adam optimizer with standard parameters, for 20 epochs with batch size 8 and with learning rate 10^{-4}. It is worth noting that DenseDepth predicts outputs at half the input resolution, but evaluates at full resolution after bilinearly upsampling.

5.4.2 Efficiency vs. Accuracy Trade-off Analysis

In this section, we study the relation between accuracy, sparsity, and efficiency of Wavelet-MonoDepth. For each set of experiments, we compare our method to an equivalently trained model without wavelets. We first study how wavelets contribute to high-frequency details, then show that they are sparse. Finally, we discuss how we trade off accuracy against efficiency by varying the threshold η used in Algorithm 1 to filter out close-to-zero coefficients.

Wavelets enhance high-frequency details. As mentioned in Section 5.3.2, the wavelet representation of depth maps allows us to output depth at different resolutions, depending on how many levels of coefficients have been computed. Tables 5.1 and 5.2 demonstrate evaluation scores for depth maps produced at different levels of wavelet decomposition on the KITTI and NYUv2 datasets respectively. As can be seen, most of the signal is captured in low-frequency estimates of the depth map at the lowest resolution. This confirms previous works observations (Chen et al. (2019a); Eigen and Fergus (2015)) that a coarse estimate of depth is sufficient to capture the global geometry of the scene.

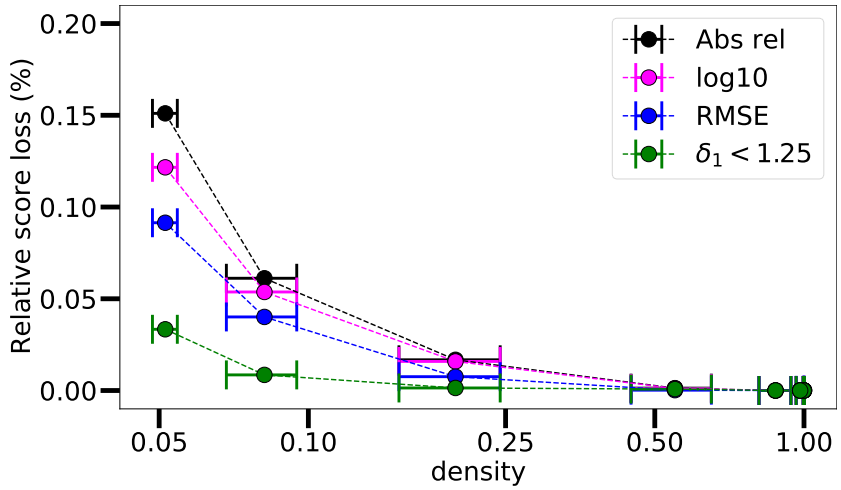


Fig. 5.5 Analysis of performance loss vs density on NYUv2. Using our wavelet representation, we can drop up to 95% of the wavelet coefficients while suffering a maximum relative performance loss of less than 0.2%.

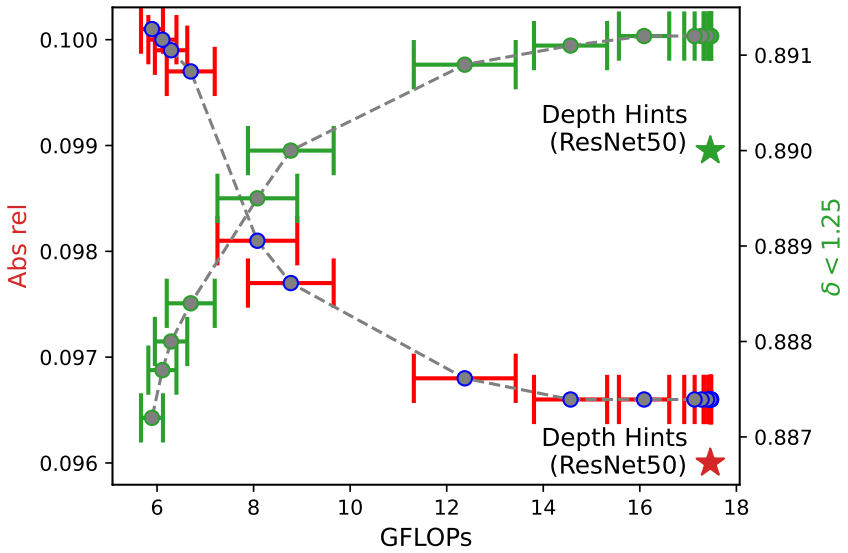


Fig. 5.6 Analysis of performance vs decoder GFLOPs on KITTI. By adjusting the parameter η, we trade off computation in GFLOPs (x-axis) against accuracy (y-axis; Abs Rel in red, and δ_1 in green). We show here that we can reduce the computation by more than half while still remaining on par with our baseline.

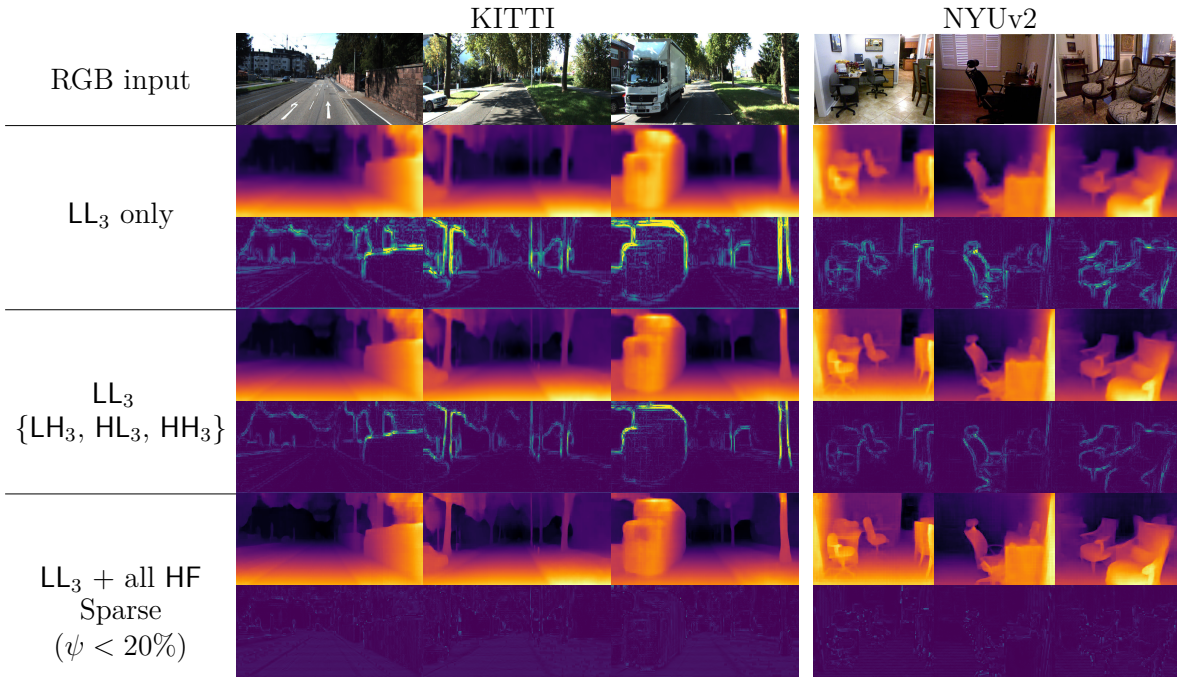


Fig. 5.7 Qualitative results on wavelet representation of depth maps. When using only a subset of wavelet scales, we run the inverse wavelet transform up to the highest scale with those coefficients, then perform a bilinear upsampling up to the full resolution. For each experiment, the bottom line shows the ℓ_1 error map between the considered depth maps and the depth map reconstructed with the complete (dense) set of predicted wavelet coefficients, which shows that wavelets contribute to refining details.

Activated HF	Abs_{Rel}	Sq_{Rel}	R	Rlog	δ_1	δ_2	δ_3
LL only	0.104	0.668	4.415	0.179	0.878	0.962	0.985
[3]	0.097	0.659	4.321	0.177	0.887	0.964	0.984
[3, 2]	0.096	0.679	4.333	0.179	0.890	0.963	0.983
[3, 2, 1]	0.096	0.702	4.366	0.180	0.891	0.963	0.983
[3, 2, 1, 0]	0.097	0.714	4.386	0.181	0.891	0.963	0.983

Table 5.1 Ablation study on high frequency coefficients on KITTI. While most of the relevant depth information is captured by the low-frequency estimate, predicting higher frequency coefficients increases accuracy. Results are evaluated without post-processing.

Activated HF	Depth Accuracy						Occ. Boundaries	
	Abs_{Rel}	RMSE	$\log_{10}$	δ_1	δ_2	δ_3	ϵ_{acc}	ϵ_{comp}
LL only	0.1281	0.5549	0.0548	0.8419	0.9674	0.9915	8.3672	9.8552
[3]	0.1264	0.5517	0.0543	0.8446	0.9680	0.9917	3.3945	8.7933
[3, 2]	0.1259	0.5512	0.0542	0.8451	0.9682	0.9917	2.1259	7.6702
[3, 2, 1]	0.1258	0.5515	0.0542	0.8451	0.9681	0.9917	1.8070	7.1073

Table 5.2 Ablation study on high frequency coefficients on NYU. While most of the relevant depth information is captured by the low-frequency estimate, predicting higher frequency coefficients increases depth and occlusion boundaries accuracy. We evaluate occlusion boundary quality using metrics from Koch et al. (2018, 2020) and the NYU-OC++ dataset (Ramamonjisoa et al. (2020); Ramamonjisoa and Lepetit (2019)).

Using more wavelet levels adds more high-frequency details to the depth map, yielding sharper results. Figure 5.7 shows the sharpening effect of wavelets qualitatively on KITTI and NYUv2 images.

Wavelets are sparse. Next, we show that high-frequency coefficients are sparse. As an example, Figure 5.1(b) shows one low-frequency and three high-frequency coefficient maps for a given depth map. We observe that the high-frequency maps have non-zero values near depth edges. More wavelet predictions can be found in supplementary. As depth edges are sparse, high-frequency coefficients at only a few pixel locations are necessary to produce high-accuracy depth maps.

Trading off accuracy against efficiency using sparsity. After training our network with standard convolutions, these are replaced with sparse ones as in Figure 5.3 and Algorithm 1. Varying the threshold value η allows us to vary the sparsity level ψ in

Equation (5.3), and consequently to trade off accuracy against complexity. Because wavelets are sparse, we can compute them only at a very small number of pixel locations and suffer a minimal loss in depth accuracy. Figures 5.4 and 5.5 show relative score changes with varying sparsity threshold on KITTI and NYU datasets respectively. Note that a fixed value of η produces different sparsity levels depending on the content of an image, so we also plot standard deviation of sparsity levels for each η value. Figure 5.4 indicates that computing the wavelet coefficients at only 10 percent of pixel locations results in a relative loss in scores of less than 1.4% for KITTI images. Similarly, Figure 5.5 shows that we can compute wavelet coefficients at only 5 percent of pixel locations while suffering a loss in scores of less than 0.20% for NYU images.

Finally, we demonstrate how sparsity of high-frequency coefficient maps can be exploited for efficiency gains in the decoder. Figure 5.6 shows Abs Rel and δ_1 scores for varying η used during prediction. As can be seen, the score change is minimal when using half multiply-add operations in the decoder and the performance is comparable to SOTA methods using only a third of multiply-add operations. Note that biggest efficiency gains are obtained at higher resolution, as sparsity increases with resolution.

5.4.3 KITTI results

We summarize our results on the KITTI dataset in Table 5.3. Here we show that our method, which simply replaces depth or disparity predictions with wavelet predictions, can be applied to a wide range of single image depth estimation models and losses. In each section of the table, the off-the-shelf model numbers are reported, together with numbers from a model trained with our wavelet formulation. For example, we demonstrate that wavelets can be used in self-supervised depth estimation frameworks such as Monodepth2 (Godard et al. (2019)), as well as its weakly-supervised extension Depth Hints (Watson et al. (2019)). We note that we achieve our best results when using Depth Hints and high-resolution input images. This is not surprising, as supervision from SGM should give better scores, but more importantly using high resolution inputs and outputs allows for more sparsification, as edge pixels become sparser as resolution grows. Importantly, we show overall that replacing fully convolutional layers with wavelets gives models with comparable performance to the off-the-shelf, non-wavelet baselines. We show qualitative results from KITTI in Figure 5.7 (left).

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
Godard et al. (2019)	Monodepth2 Resnet18	✓	S	192 × 640	0.108	0.842	4.891	0.207	0.866	0.949	0.976
	WaveletMonodepth Resnet18	✓	S	192 × 640	0.110	0.876	4.916	0.206	0.864	0.950	0.976
	Monodepth2 Resnet50	✓	S	192 × 640	0.108	0.802	4.577	0.185	0.886	0.963	0.983
	WaveletMonodepth Resnet50	✓	S	192 × 640	0.106	0.824	4.824	0.205	0.870	0.949	0.975
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	192 × 640	0.106	0.780	4.695	0.193	0.875	0.958	0.980
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.106	0.813	4.693	0.193	0.876	0.957	0.980
	Depth Hints Resnet50	✓	S_{SGM}	192 × 640	0.102	0.762	4.602	0.189	0.880	0.960	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.105	0.813	4.625	0.191	0.879	0.959	0.981
Godard et al. (2019)	Monodepth2 Resnet18	✓	MS	192 × 640	0.104	0.786	4.687	0.194	0.876	0.958	0.980
	WaveletMonodepth Resnet18	✓	MS	192 × 640	0.109	0.814	4.808	0.198	0.868	0.955	0.980
Watson et al. (2019)	Depth Hints	✓	MS + S_{SGM}	192 × 640	0.105	0.769	4.627	0.189	0.875	0.959	0.982
	WaveletMonodepth Resnet18	✓	MS + S_{SGM}	192 × 640	0.110	0.840	4.741	0.195	0.868	0.956	0.981
Godard et al. (2019)	Monodepth2 Resnet18	✓	S	320 × 1024	0.105	0.822	4.692	0.199	0.876	0.954	0.977
	WaveletMonodepth Resnet18	✓	S	320 × 1024	0.105	0.797	4.732	0.203	0.869	0.952	0.977
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	320 × 1024	0.099	0.723	4.445	0.187	0.886	0.961	0.982
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.102	0.739	4.452	0.188	0.883	0.960	0.981
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.096**	**0.710**	4.393	0.185	0.890	**0.962**	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	0.097	0.718	**4.387**	**0.184**	**0.891**	**0.962**	**0.982**
	WaveletMonodepth Resnet50 ($\eta = 0.05$)	✓	S_{SGM}	320 × 1024	0.100	0.726	4.444	0.186	0.888	**0.962**	**0.982**

Table 5.3 Quantitative results on KITTI. We compare our method to our baselines on KITTI Geiger et al. (2012), using the Eigen split. The *Data* column indicates the training data modality: S is for self-supervised training on stereo images, MS is for models trained with both monocular (forward and backward frames) and stereo data and S_{SGM} refers to the extra stereo ground truth which was used in Watson et al. (2019).

Method	H × W	Abs Rel	RMSE	log_{10}	δ<1.25	δ<1.25^2	δ<1.25^3	ϵ_{acc}	ϵ_{comp}
DenseNet baseline	480 × 640	0.1277	**0.5479**	**0.0539**	0.8430	**0.9681**	**0.9917**	**1.7170**	**7.0638**
Ours (last scale sup.)	480 × 640	0.1280	0.5589	0.0546	0.8436	0.9658	0.9908	1.7678	7.1433
Ours	480 × 640	**0.1258**	0.5515	0.0542	**0.8451**	**0.9681**	**0.9917**	1.8070	7.1073
Ours ($\eta = 0.04$)	480 × 640	0.1259	0.5517	0.0543	0.8450	**0.9681**	**0.9917**	1.8790	7.0746

Table 5.4 Quantitative results on NYUv2 (Silberman et al. (2012)) We compare our DenseDepth (Alhashim and Wonka (2018))-inspired baseline to our implementation with wavelets and with sparsity. All results are evaluated in the Eigen center crop, without post-processing. As in DenseDepth, our network outputs a 240 × 320 depth map which is then upsampled for evaluation.

5.4.4 NYUv2 results

Scores on NYUv2 are shown in Table 5.4. Our method performs on par with our baseline, which demonstrates that it is possible to estimate accurate depth and sparse wavelets without directly supervising the wavelet coefficients, in contrast with Yang et al. (2020). In Table 5.4, we show that supervising depth only at the last scale performs on par with our network supervised at all scales, which shows that a full multi-scale wavelet reconstruction network can be trained end-to-end. Qualitative results from NYUv2 are shown in Figure 5.7 (right).

5.5 Conclusion

In this work we combine wavelet representation with deep learning for a single-image depth prediction task. We demonstrate that a neural network can learn to predict wavelet coefficient maps through supervision of the reconstructed depth map with existing losses. Our experiments using KITTI and NYUv2 datasets show that we can achieve scores comparable to SOTA models using similar encoder-decoder neural network architectures to the baseline models, but with wavelet representations.

We also analyze sparsity of wavelet coefficients and show that sparsified wavelet coefficient maps can generate high-quality depth maps. Finally, we exploit this sparsity to reduce multiply-add operations in the decoder network by at least a factor of 2.

Acknowledgements

We would like to thank Aron Monszpart for helping us set up cloud experiments, and our reviewers for their useful suggestions.

Chapter 6

Reconstructing 3D Scenes as Complex Sets of Primitives

The work described in this chapter is based on the following publication: MonteBoxFinder: Detecting and Filtering Primitives to Fit a Noisy Point Cloud, published at ECCV 2022.

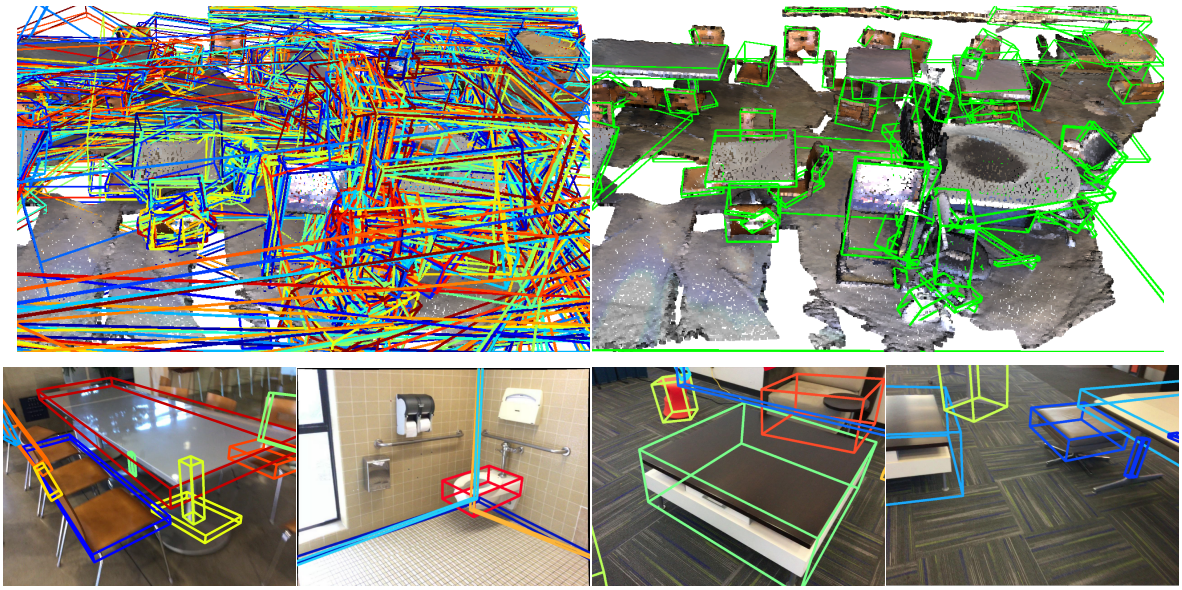


Fig. 6.1 Given a noisy 3D scan with missing data, our method extracts many possible cuboids, and then efficiently selects the subset that fits the scan best. The resulting cuboids can then be projected onto each camera viewpoint (bottom row), then used as ground truth to supervise single image cuboid prediction methods.

Abstract

We present MonteBoxFinder, a method that, given a noisy input point cloud, fits cuboids to the input scene. Our primary contribution is a discrete optimization algorithm that, from a dense set of initially detected cuboids, is able to efficiently filter good boxes from the noisy ones. Inspired by recent applications of MCTS to scene understanding problems, we develop a stochastic algorithm that is, by design, more efficient for our task. Indeed, the quality of a fit for a cuboid arrangement is invariant to the order in which the cuboids are added into the scene. We develop several search baselines for our problem and demonstrate, on the ScanNet dataset, that our approach is more efficient and precise. Finally, we strongly believe that our core algorithm is very general and that it could be extended to many other problems in 3D scene understanding.

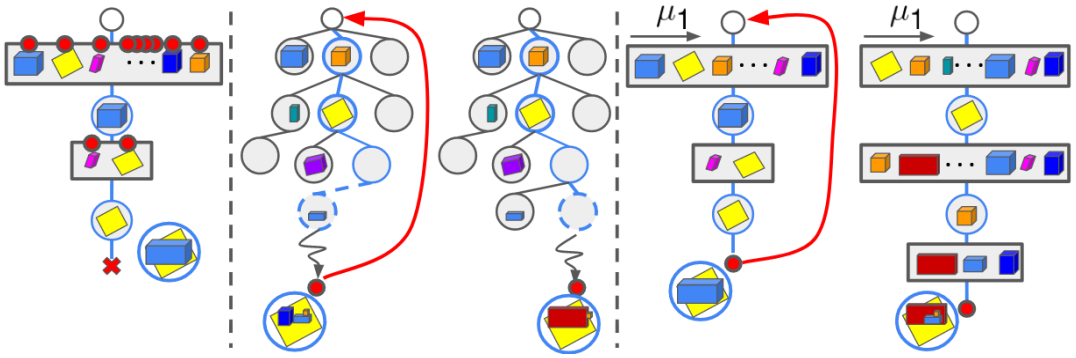
6.1 Introduction

Previous chapters (3-4-5) presented ways to improve single image depth estimation algorithms. While depth maps are a dense representation of 3D from a single view point, we are also interested in being able to represent a 3D scene with a set of simple geometric primitives. The latter task is a long-standing computer vision problem (Roberts (1963)). Solving it would provide a light representation of 3D scenes that is arguably easier to exploit by many downstream applications than a depth map or a 3D point cloud for example. But maybe more importantly, this would also demonstrate the ability to reach a "high-level understanding" of the scene's geometry, by creating a drastically simplified representation. The following work presents a first step to single-image cuboid detection, as reliable cuboids could then prove useful to supervise cuboid estimation from *single image* methods, by projecting these cuboids into the original camera views (see Figure 6.1). Training such methods is left for future work.

In this work, we start from a point cloud of a indoor scene, which can be obtained by 3D reconstruction from images or scanning with an RGB-D camera. Recent works have considered representing 3D point clouds with primitives (Deprelle et al. (2019); Groueix et al. (2018); Paschalidou et al. (2019, 2020)); however they consider "ideal 3D input data", in the sense that the point cloud is complete and noise-free. By contrast, point clouds from 3D reconstruction or scans are typically very noisy with missing data, and robust methods are required to handle this real data.

To be robust to noise and missing data, we propose a discrete optimization-based method. Our approach does not require any training data, which would be very cumbersome to create manually. Given a point cloud, we extract a large number of primitives. While in our experiments we consider only cuboids as our primitives, our approach can be generalized to other choices of primitives. We rely on a simple *ad hoc* algorithm (Schnabel et al. (2007)) to obtain an initial set of primitives. We expect this algorithm to generate correct primitives but also many false positives. Our problem then becomes the identification of the correct primitives while rejecting the incorrect ones, by searching the subset of primitives that explains the scene point cloud the best.

While the theoretical combinatorics of this search are huge, as they grow exponentially with the number of extracted primitives, the search is structured by some constraints. For example two primitives should not intersect. To tackle this problem, we take inspiration from a recent work on 3D scene understanding (Hampali et al. (2021)). Hampali et al. (2021) proposes to rely on the Monte Carlo Tree Search (MCTS) algorithm to handle a similar combinatorial problem to select objects' 3D models: The MCTS algorithm is probably best known as the algorithm used by AlphaGo (Silver et al. (2016)). It is



Hill-Climbing 2 iterations of MCTS 2 iterations of our algorithm

Fig. 6.2 Comparative overview The hill-climbing algorithm—simply taking the primitive that improves the most the objective function— can terminate × quickly as it gets stuck into a local minimum because of the constraints between primitives. MCTS as used in Hampali et al. (2021) explores iteratively the solution tree by traversing blue paths, updating which primitives are the most promising ones, but keeping the tree structure fixed. At each iteration, our approach also updates (→) which primitives are the most promising ones, and starts with them. This makes our approach identify a good solution much faster than MCTS in general. Red circles ● represent objective function evaluations. Hill-climbing has to evaluate the complete objective function each time it considers a primitive, while MCTS and our algorithm evaluate the objective function only at the end of an iteration when a complete solution is complete.

typically used to explore the tree of possible moves in the game Go because it scales particularly well to high combinatorics. Hampali et al. (2021) adapts it to 3D models selection by considering a move as the selection of a 3D model for one object, and showed it performs significantly better than the simple hill-climbing algorithm that is sometimes used for similar problems (Zou et al. (2019)). Another advantage of this approach is that it does not impose assumptions on the form of the objective function, unlike other approaches based on graphs, for example Shao et al. (2014).

While exploring the solution tree with MCTS as done in Hampali et al. (2021) is efficient, we show we can still speed up the search for a solution significantly more. The tree structure imposes an ordering of the possible 3D models to pick from. Such sequential structures are necessary when MCTS is applied to games as game moves depend on the previous ones, but we argue that there is a more efficient alternative in the case of object detection and selection for scene understanding.

As illustrated in Figure 6.2, MCTS works by performing multiple iterations over the tree structure, focusing on the most promising moves. The estimate of how much a move is promising is updated at each iteration. For our problem of primitive selection, we

propose to also proceed by iteration. Instead of considering a tree search, at the end of each iteration, we sort the primitives according to how likely they are to belong to the correct solution. The next iteration will thus evaluate a solution that integrates the most promising primitives. Our experiments show that this converges much faster to a correct solution.

To evaluate our approach, we experiment on the ScanNet dataset (Dai et al. (2017a)), a large and challenging set of indoor 3D RGB-D scans. It contains 3D point clouds of real scenes, with noisy captures and large missing parts, as some parts were not scanned and dark or specular materials are not well captured by the RGB-D cameras. We did not find any previous work working on similar problems, but we adapted other algorithms, namely a simple hill-climbing approach (Zou et al. (2019)) and the MCTS algorithm of Hampali et al. (2021) to serve as our baselines for comparison. To do so, we introduce several metrics to evaluate the fit quality.

Our algorithm is conceptually simple, and can be written in a few lines of pseudo-code. We believe it is much more general than the cuboid fitting problem. It could first be extended to other type of primitives, and applied to many other selection problems with high combinatorics, and could be applied to other 3D scene understanding problems, for auto-labelling for example. We hope it will inspire other researchers for their own problems

6.2 Related Work

In this section, we first discuss related work on cuboid fitting, and then on possible optimisation methods to solve our selection problem.

6.2.1 Cuboid Fitting on Point Clouds

Primitive fitting is a long standing Computer Vision problem. In the section, we only discuss about methods that operate on point clouds, although there are a large number of methods that are seeking progress in the field of cuboid fitting from 2D RGB images (Gupta et al. (2010); Kluger et al. (2021); Roberts (1963)).

Object scale. Sung et al. (2015) leveraged cuboids decompositions to improve 3D object completion of scans of synthetic objects. Tulsiani et al. (2017) introduced object abstraction using cuboids on more challenging objects from the Shapenet (Chang et al. (2015)) dataset. Paschalidou et al. (2019) extended Tulsiani et al. (2017) by using the

more expressive superquadrics to fit 3D objects. However these methods only operate at the scale of a single objects, on synthetic data, and always assume or are limited to a moderate number of primitives. Some older work related to us have focused on parsing an input point cloud as a decomposition into primitives. Li et al. (2011) decompose a *real* scan of an object into primitives by extracting a set of primitives with RANSAC, which they refine by reasoning on relationship between these primitives. However their method works only on very clean scans, and using object that were *built* as a set of primitives. Furthermore, since they reason about interaction between primitives using a graph, the complexity of there method quickly becomes untractable.

Room-scale cuboid detection. Another class of works has focused on room-scale 3D point cloud parsing with cuboids. A large number of works focused on detecting object bounding boxes in 3D scans have recently emerged since the deep learning era (Qi et al. (2019); Shi et al. (2020); Shi and Rajkumar (2020)). Guo et al. (2020) wrote a great survey regarding these methods. Contrary to these methods, our method is able to parse 3D scans with cuboids at the granularity level of parts of objects. Jiang and Xiao (2013) used RGB-D images to fit cuboids to the point cloud obtained by the depth map. In contrast to us, they operate using single-view images, but also leverage color cues via superpixels. Shao et al. (2014) also parse depth maps with cuboids. Given an initial set of cuboids, they build a graph to exploit physical constraints between them to refine the cuboids arrangement. However, they still require human-in-the-loop for challenging scenes, and their graph based method limits the number of cuboids that can be retrieved without exceeding complexity. Our method, in contrast, can deal with number of cuboids that are an order of magnitude larger.

6.2.2 Solution Search for Scene Understanding

We focus here on scene understanding methods which, like us, do not rely on supervised training data for complete scenes, even if some of them require training data to recognize the objects. These methods typically start from a set of possible hypotheses for the objects present in the scene (similar to the primitives in our case), and choose the correct ones with some optimization algorithms.

Monte Carlo Markov Chain (MCMC) Andrieu et al. (2003) is a popular algorithm to select the correct objects in a scene by imposing constraints on their arrangement. MCMCs can be applied to a parse graph (Chen et al. (2019b); Choi et al. (2013); Huang et al. (2018b); Zhao and Zhu (2013)) that defines constraints between objects. However,

Method	Uphill	MCTS	Ours
Exploratory	✗	✓	✓
Stochastic	✗	✓	✓
Leverage order invariance	✓	✗	✓

Table 6.1 Properties of different solution search methods. Our method leverages all popular mechanisms for efficient solution search while leveraging the structure of the problem, which does not require employing tree structures for solution search.

this parse graph needs to be defined manually or learned from manual annotations. Also, MCMCs typically converge very slowly.

Greedy approaches were also used in previous works (Izadinia et al. (2017)), and they rely on a hill-climbing method to find the objects' poses (Izadinia et al. (2017)). Zou et al. (2019) selects objects using hill-climbing as well by starting from the objects with the best fits to an RGB-D image. While simple and greedy, this approach can work well on simple scenes. However, it can easily get stuck on complex situations, as our experiments show. Lee et al. (2010) uses beam search but this is also an approximation as it also cuts some hypotheses to speed up the search.

Monte Carlo Tree Search (MCTS) was recently used in Hampali et al. (2021), where they proposed to use MCTS as an optimization algorithm to choose objects that explain an RGB-D sequence. Hampali et al. (2021) adapts MCTS by considering the selection of one object as a possible move in a game. The moves are selected to optimize an objective function based on the semantic segmentation of the images and the depth maps. The advantage of this approach is that MCTS can scale to complex scenes, while optimizing a complex objective function.

Our approach is motivated by Hampali et al. (2021). However, we generate the primitives in a very different way, but more importantly, we propose a novel optimization algorithm, which, contrary to MCTS, does not rely on a tree structure, making it is simpler and significantly more efficient than MCTS, as demonstrated by our experiments.

6.3 Method

In this section, we first describe how we extract a large pool of cuboids from a given 3D scan. Then, we formalize the selection of the optimal cuboid arrangement. Finally, we detail the solution we propose.

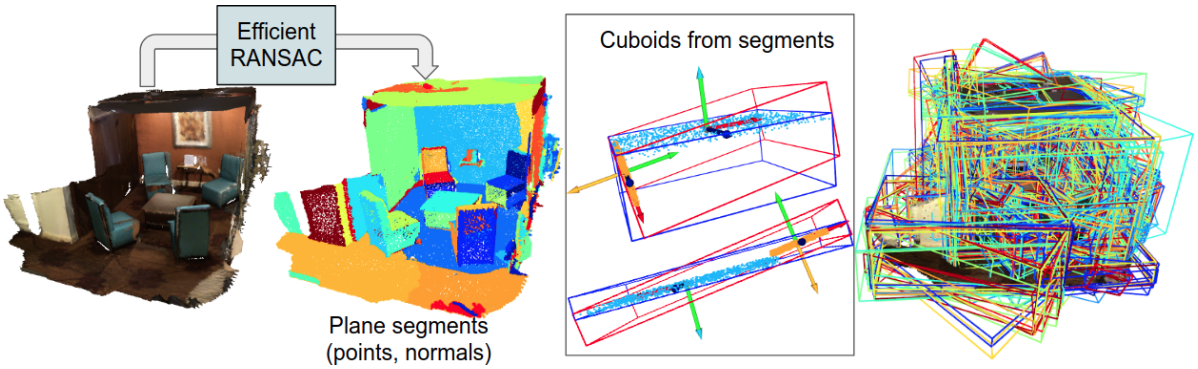


Fig. 6.3 Overview of our cuboid generation pipeline. After extracting plane segments using an off-the-shelf algorithm Schnabel et al. (2007), we construct cuboids around these segments and pairs of adjacent segments. The result is a dense set of cuboids, which may contain many false positives.

6.3.1 Generating Cuboid Proposals from Noisy Scans

Figure 6.3 summarizes our cuboid proposal generation pipeline. The goal of this pipeline is to provide a large pool of cuboids. Some extracted cuboids can be false positives at this stage. The correct subset of cuboids will be selected by the next stage. In this way, we can be robust to noise and missing data in the 3D scan. Our pipeline can be divided in 3 steps: (1) we first extract plane segments; (2) we construct cuboids from pairs of plane segments; (3) we also construct *thin* cuboids by fitting a 3D bounding box to the each plane segment individually. These thin cuboids allow us to represent planar surfaces as well in the final representation. On average, we obtain 880 cuboids and 174 *thin* cuboids per scene.

Extracting planes segments. We use Efficient-RANSAC by Schnabel et al. (2007) to extract 3D planes from the input point cloud. Efficient-RANSAC identifies and returns planar connected components made of 3D points. It is controlled by three hyperparameters: a threshold on the plane-to-point distance to count the inliers, a threshold on the cosine-similarity between normals to points, and a connectivity radius. We use the same hyperparameters for all the scenes in ScanNet, although we could run RANSAC multiple times with various geometric parameters in order to adapt to various types of noise, and still be able to efficiently filter out false positives.

Constructing boxes from pairs of planes. Given a set of planes segments $\{\pi_i = (X_i, \boldsymbol{N}_i)\}$, where a plane segment π is represented as a point cloud X and its fitted plane normal $\boldsymbol{N}$, we construct bounding boxes from all pairs of planes (π_A, π_B) that satisfy

two criteria, *alignment* and *proximity*. Alignment means that the two normals should be orthogonal or co-linear. Proximity enforces planes segments to have at least one connected component in 3D. We then employ two Gram-Schmidt ortho-normalizations to obtain the frame coordinate of two bounding boxes, which are computed to enclose $X_A \cup X_B$. More details can be found in the supplementary material.

Fitting 3D bounding boxes to 3D plane segments. Since we want our method to also retrieve thin objects that may not have compatible neighbors, we therefore fit a 3D oriented bounding box to each plane segment's point cloud X, using the efficient "*Oriented Bounding Box*" method from Chang et al. (2011).[1]

6.3.2 The Cuboids Arrangement Search Problem

We now want to select a subset $\mathcal{S}'$ of $\mathcal{S}$, the set of cuboids generated in Section 6.3.1, which fits well the input point cloud X of the scene. The cuboids in $\mathcal{S}'$ should not mutually intersect to ensure a minimal representation of this scene.

To solve this problem, we consider (1) an objective function ℓ, defined in Algorithm 2, which will guide the search towards the best solution, and (2) a search algorithm such as the baselines described in Section 6.3.3, that should be designed to converge to the best solution as efficiently as possible. To better present the algorithms, we introduce a `Cuboid` Class, which we present first.

`Cuboid` Class. We define a `Cuboid` class to instantiate cuboids for our solution search algorithms. It is described by its faces normals and its 8 corners, yielding a surface mesh from which we can sample 3D points. Other attributes can be added to a `Cuboid` , depending on the needs of a particular algorithm, e.g. the number of times a `Cuboid` s has been used in a solution can be denoted as $s.n_1$.

To enforce constraints between cuboids, we need to test if the intersection between two cuboids is small enough. We define this criterion using a variation of the measure of a Intersection-over-Union criterion, and provide its pseudo-code in Supp Mat. `isCompatible`(s_1, s_2, η) measures the ratio between the volume $vol(s_1 \cap s_2)$ of the intersection between both cuboids s_1 and s_2, and the minimum of the volumes of each cuboid $vol(s_1)$ and $vol(s_2)$. In practice, we approximate these volumes by uniformly randomly sampling points from both cuboids and count the points that are inside both s_1 and s_2. The volume ratio is then compared to a threshold η, to decide if the two `Cuboid`

[1] We used CGAL's (The CGAL Project (2022)) implementation of Schnabel et al. (2007) and Chang et al. (2011).

Algorithm 2: Loss function ●

procedure `evalObjFunc`$(S,\ Y)$

Input : Set of cuboids S, target point cloud Y and its normals $\boldsymbol{N}(Y)$;

$(X, \boldsymbol{N}(X)) \leftarrow$ sample_mesh_surface(S);

$\ell_c :=$ ChamferDistance$(X \rightarrow Y)$ + ChamferDistance$(Y \rightarrow X)$;

$\ell_n :=$

CosineDissimilarity$(\boldsymbol{N}(X) \rightarrow \boldsymbol{N}(Y))$ + CosineDissimilarity$(\boldsymbol{N}(Y) \rightarrow \boldsymbol{N}(X))$;

return $\ell_c \cdot (1 + 0.25 \cdot \exp(\ell_n))$;

intersect. While this test can be performed "*on the fly*" when searching solutions, we pre-compute the pair-wise `Cuboid` compatibility matrix in advance for efficiency.

Objective Function We aim to minimize the distance between our cuboids and the target point cloud, while keeping its normals aligned with the pointcloud's normals. We use Chamfer Distance (CD) and Cosine Dissimilarity, *i.e.* the complement of Cosine Similarity, as our distance and normals deviation losses, yielding full objective function is described in Algorithm 2. In the loss, we truncate CD to $\tau = 0.1$, and normalize it by τ.

6.3.3 Solution Search Baseline Algorithms

6.3.3.1 Hill-Climbing Algorithm.

The first baseline for our discrete optimization problem is the Hill-Climbing algorithm (Skiena (2010)), a naive greedy descent algorithm. This algorithm constructs a solution iteratively, where at each iteration, it comprehensively searches for the proposal that best improves the loss function of a solution $\mathcal{S}_F$, while leaving the solution valid *i.e.* with no incompatibilities. If no proposal is available nor can improve the objective function, the algorithm stops ×. The pseudo code for Hill-Climbing is given in the supplementary material.

6.3.3.2 MCTS Algorithm.

We first describe here the MCTS algorithm, as it inspired our algorithm. Browne et al. (2012) provides a full description of the MCTS algorithm. We present it in the context of our cuboid selection problem, following what was done in Hampali et al. (2021) for 3D model selection. Hampali et al. (2021) provides a pseudo code for MCTS.

MCTS is able to efficiently explore the large trees that result from the high combinatorics of some games such as Go. As represented in Figure 6.2, the nodes of the tree

correspond to possible states, and the branches to possible moves. MCTS does not build explicitly the entire tree—this would not be tractable anyway—, but only a portion of it, starting from the root at the top.

⇝**Simulation step.** Nodes are thus created progressively at each iteration. To decide which nodes should be created, the existing nodes contain in addition to a state an estimate V of the *value* of this state. To initialize V, MCTS uses a *simulation step* denoted ⇝ in Figure 6.2, which explores randomly the rest of the tree until reaching a leaf without having to build the tree explicitly. For games, reaching a leaf corresponds to either winning or loosing the game. If the game is won, V should be large; if the game is lost, V should be small.

Adaptation to our problem. Figure 6.2 shows that in our case, a state in a node is the set of primitives that have been selected so far. A "move" corresponds to adding a primitive to the selected primitives. The children of a node contain primitives that are mutually incompatible, and compatible with the primitives in the ancestor nodes: Such structure ensures that every path in the tree represents a valid solution. In this paper, we consider two possibilities: A varying number of children as in Hampali et al. (2021) and MCTS-Binary, a binary tree version of MCTS: In MCTS-Binary, a node has two children, corresponding to selecting or skipping a primitive. More details are provided in the supplementary material.

×"Reaching a leaf" happens when no more primitives can be added, because we ran out of primitives or because all the remaining primitives intersect with the primitives already selected. The value V of the new nodes are initialized after the simulation step by evaluating the objective function ●for the set of primitives for the leaf. We take this objective function as a fitness measure between the primitives and the point cloud. Note that this function does not need to have special properties, nor do we need heuristics to guide the tree search.

Selection and expansion steps. At each iteration, MCTS traverses the tree starting from the root node, often using the standard Upper Confidence Bound (UCB) criterion Browne et al. (2012) to choose which branch to follow. A high UCB score for a node means that it is more likely to be part of the correct solution. This criterion depends on the values V stored in the nodes and balances exploitation and exploration: When at a node N, we continue with its child node N' that maximizes the UCB score, which depends on the number of times N and N' have been visited so far. This criterion allows MCTS to balance exploration and exploitation.

At some point of this traversal procedure, we will encounter a node with a child node N that has not been created yet, we add the child node to the tree. We use the simulation step described above to initialize $V(N)$ and initialize $n(N)$ to 1.

→**Update step.** MCTS also uses the value $V(N)$ to improve the value estimate of each node N' visited during the tree traversal. Different ways to do so are possible, and we found that for our problem, it is better to take the maximum between the current estimate $V(N')$ and $V(N)$: $V(N') \leftarrow \max(V(N'), V(N))$. $n(N')$, the number of times the node was visited is also incremented.

Final solution. After a chosen number of iterations, MCTS stops. For our problem, we obtain a set of primitives by doing a tree traversal starting from the root node and following the nodes with the highest values V.

6.3.4 Our algorithm: MonteBoxFinder

We first review the issues when using MCTS for our problem, then give an overview of our algorithm and its components. Finally, we provide some details for each component.

6.3.4.1 Moving from MCTS.

Our primitives selection algorithm is inspired by MCTS, and it is motivated by two observations that show that MCTS is not optimal for our selection problem:

- the order we select the primitives does not matter. However, MCTS keeps growing its tree without modifying the nodes already created. This implies that if a primitive appears at the top of the tree but does not actually belong to the correct solution, it will slow down the convergence of MCTS towards this solution.
- if a node corresponding to adding some primitive P has a high value V, the node corresponding to not keeping P should have a low value, and vice versa. There is no mechanism in MCTS as used in Hampali et al. (2021) to ensure this. This is unfortunate as one iteration could be used to update more nodes than only the visited nodes.

6.3.4.2 Overview.

We give an overview of our algorithm in Algorithm 3. To exploit the two observations described above, we do not use a tree structure. Instead, we use the list of primitives

Algorithm 3: Our MonteBoxFinder Algorithm

Result: Set of selected `Cuboid` $\mathcal{S}_F$
Input : Set of available `Cuboid` $\mathcal{S}$;
Number of evaluations N_{eval} ;
Threshold η ;
Current solution $\mathcal{S}_c := \emptyset$;
Final solution $\mathcal{S}_F := \emptyset$;
Current best loss $\ell^* := +\infty$;
procedure `InitializeNodes(`$\mathcal{S}$`)`
  Input : Pool of `Cuboid` $\mathcal{S}$;
  $\mathcal{S} \leftarrow$ Shuffle$(s \in \mathcal{S})$;
  $\mathcal{S}_c \leftarrow$`Simulate(`$\mathcal{S}$, η`)`;
  $\ell \leftarrow$ `evalObjFunc(`$\mathcal{S}_c$`)`;
  `// Update ALL Cuboid states`
  $\mathcal{S} \leftarrow$ `Update(`$\mathcal{S}$, $\mathcal{S}_c$, ℓ`)`;
  return $\mathcal{S}$

`// MonteBoxFinder Core Algorithm`
$\mathcal{S} \leftarrow$ `InitializeNodes(`$\mathcal{S}$`)`;
for (iter=0; iter$\neq N_{eval}$; iter++) {
  $\mathcal{S} \leftarrow \text{Sorted}_{\downarrow}(s \in \mathcal{S}, s \mapsto s.\mu_1)$;
  $\mathcal{S}_c \leftarrow$`Simulate(`$\mathcal{S}$, η`)`;
  $\ell \leftarrow$ `evalObjFunc(`$\mathcal{S}_c$`)`;
  `// Update ALL Cuboid states`
  $\mathcal{S} \leftarrow$ `Update(`$\mathcal{S}$, $\mathcal{S}_c$, ℓ`)`;
  if $\ell < \ell^*$ **then**
    $\ell^* \leftarrow \ell$;
    $\mathcal{S}_F \leftarrow \mathcal{S}_c$;
}
return Best solution $\mathcal{S}_F$;

which we sort at each iteration, by exploiting our current estimate for each primitive to be part of the current solution. Our method progressively estimates and exploits a prior probability $\mathcal{P}$ for a primitive to belong to the solution based on our adaptation of the Upper Bounding Criterion (UCB) that balances the exploitation vs. exploration trade-off.

6.3.4.3 Initialization.

We initialize the run with a few random traversals in order to initialize the states of each `Cuboid` proposal.

6.3.4.4 Simulate. ($\rightsquigarrow$)

At every iteration we first sort primitives $\mathcal{S}_A$ according to their confidence value $s.\mu_1$ in descending order, hence more confident primitives will be more likely selected. Afterwards, we perform the simulation that pops primitives s from sorted $\mathcal{S}_A$. With probability $\mathcal{P}_\epsilon = 0.3$, we perform exploitation and add s to the list of selected proposals $\mathcal{S}_F$ if $(s.\mu_1 > s.\mu_0)$. Otherwise, we perform exploration and add s to $\mathcal{S}_F$ if $(s.\mu_1 < s.\mu_0)$.

Algorithm 4: Simulate($\rightsquigarrow$) and Update($\rightarrow$) functions of our algorithm

Input : Exploration probability $\mathcal{P}_\epsilon$;
Threshold δ;
procedure `Simulate`($\mathcal{S}_A$, η)
  Input : Pool of available `Cuboid` $\mathcal{S}_A$,
    threshold η
  Output $\mathcal{S}_F := \emptyset$;
  for ($s \in \mathcal{S}_A$) {
    if s.`isCompatible`($\mathcal{S}_F$, η) **then**
      $\epsilon :=$ uniform_sample($[0, 1]$);
      if ($\epsilon < \mathcal{P}_\epsilon$) **then**
        if ($s.\mu_1 > s.\mu_0$) **then**
          $\mathcal{S}_F$.add(s)
      else
        if ($s.\mu_1 < s.\mu_0$) **then**
          $\mathcal{S}_F$.add(s)
      end
  }
  return $\mathcal{S}_F$

procedure `Update`($\mathcal{S}$, $\mathcal{S}_F$, ℓ)
  Input : Full pool of `Cuboid` $\mathcal{S}$;
  Selected set of `Cuboid` $\mathcal{S}_F \subset \mathcal{S}$;
  Solution score ℓ;
  for ($s \in \mathcal{S}$) {
    if $s \in \mathcal{S}_F$ **then**
      `// Update best` ℓ `when kept`
      $s.\ell_1 \leftarrow \min(\ell, s.\ell_1)$
      $s.n_1 \leftarrow s.n_1 + 1$
      $s.\mu_1 \leftarrow -s.\ell_1 + \sqrt{\ln(1/\delta)/s.n_1}$
    else
      `// Update best` ℓ `when rejected`
      $s.\ell_0 \leftarrow \min(\ell, s.\ell_0)$
      $s.n_0 \leftarrow s.n_0 + 1$
      $s.\mu_0 \leftarrow -s.\ell_0 + \sqrt{\ln(1/\delta)/s.n_0}$
    end
  }
  return $\mathcal{S}$

6.3.4.5 UCB Criterion.

We modified the UCB score to fit our algorithm, which does not rely on a tree structure. We use this modified term to estimate two confidence measures $s.\mu_0$ and $s.\mu_1$ reflecting how much a cuboid s is likely to belong to the correct solution or not:

$$s.\mu_0 = -s.\ell_0 + \sqrt{\ln(1/\delta)/s.n_0}, \quad s.\mu_1 = -s.\ell_1 + \sqrt{\ln(1/\delta)/s.n_1}, \tag{6.1}$$

where $s.\rho_0$ and $s.\rho_1$ are the minimum loss values reached when rejecting and accepting primitive s, $s.n_0$ and $s.n_1$ denote the number of times that the primitive were rejected and selected respectively, and $\delta = 0.03$ is a hyperparameter modifying the exploration rate, smaller δ implies larger exploration.

6.3.4.6 Update ($\rightarrow$)

In comparison with the *update step* of MCTS described in 6.3.3, our MonteBoxFinder algorithm updates *all* primitives states after an iteration. If a primitive s was selected, we update its $s.\ell_1$, $s.\mu_1$, and $s.n_1$ values based on the obtained loss ℓ and our adapted UCB criterion, otherwise we update its $s.\ell_0$, $s.n_0$, and $s.\mu_0$ values instead. In the next

iteration during simulation, we use these value to determine whether to select or reject the primitive.

6.4 Experiments

6.4.1 Dataset

ScanNet (Dai et al. (2017a)) is a dataset that contains noisy 3D scans of 1613 indoor scenes. We evaluate our method on the full dataset, where for each scene, we used the decimated and cleaned point clouds provided in ScanNet both for the box proposals generation step and for the solution search step.

6.4.2 Metrics

6.4.2.1 Fitness measures.

The most direct way to measure the quality of a solution is to measure the loss function ℓ described in Algorithm 2. Indeed, we want to evaluate the ability of our algorithm to search the solution space. Additionally, we measure a bi-directional precision metric Pr_τ. $\text{Pr}_\tau$is computed as the proportion of points successfully matched between the "*synthetic*" point cloud X, generated by sampling 3D points from retrieved 3D cuboid meshes, and the 3D scan Y. A point is successfully matched if its Chamfer Distance (CD)[2] value is below a threshold $\tau = 0.2$:

$$\text{Pr}_\tau = \frac{|\{x \in X \text{ s.t. } CD[x \to Y] \leq \tau\}|}{2|X|} + \frac{|\{y \in Y \text{ s.t. } CD[y \to X] \leq \tau\}|}{2|Y|} . \tag{6.2}$$

6.4.2.2 Efficiency measure.

The motivation for developing our approach compared to Hampali et al. (2021) is to converge faster towards a good solution. In order to measure efficiency of a given method, we consider the curve of the objective function of the best found solution as a function of the iteration, as the ones showed in Figure 6.4. We use the Area Under the Curve (AUC) given a maximum budget of iterations N_{eval}: the lower the AUC, the faster the convergence. We also report AUC (norm), which normalizes the AUC values of the different between 0 and 1, with 0 being the value of the best performing method and 1 being the value of the worst performing method.

[2]In this case, we do not apply the normalization discussed in Algorithm 2

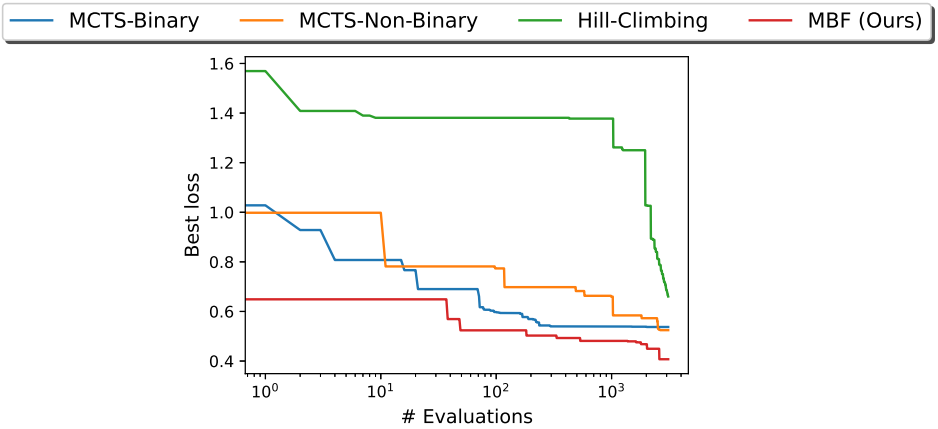


Fig. 6.4 Value of the objective function for the best found solution as a function of the number of evaluations for Hill-Climbing, MCTS, MCTS-Binary, and our MonteBoxFinder (MBF) method. Hill-Climbing requires many evaluations before finding a reasonable solution, which explains the flat curve at the beginning. It also gets stuck into a local minimum and stops improving. In this experiment, we give the number of evaluations Hill-Climbing used before getting stuck to the three other methods. Our method converges significantly faster than the other methods towards a better solution. Similar graphs for other scenes are provided in the supplementary material.

6.4.2.3 Complexity measure.

We observe that bad solutions tend to contain a small number of selected primitives. This is because it is challenging to find a large subset of cuboids with no intersection between any pair of cuboids. Hence we also report the number of cuboids in the retrieved solutions.

6.4.3 Evaluation Protocol

For all scenes from the ScanNet dataset (Dai et al. (2017a)), we run the Hill-Climbing method, and obtain its solution $\mathcal{S}_{\text{HC}}$. We then consider the number N_{eval} of evaluations of the objective that were required by Hill-Climbing to construct this solution. We then run MCTS and our algorithm using the same number of evaluations N_{eval}. This ensures the three methods are compared fairly, as they are given the same evaluation budget, which is by far the most costly step of all three algorithms.

6.4.4 Quantitative results

Table 6.2 provides the results of our experimental comparisons. As expected, the Hill-Climbing algorithm performs worst: By greedily selecting proposals that minimize the

	Loss↓	Precision ↑	AUC ↓	AUC (norm) ↓	Avg. # Cuboids ↑
Hill-Climbing	0.383	0.928	0.871	0.998	12
MCTS	0.247	0.966	0.427	0.225	28
MCTS-Binary	0.292	0.961	0.370	0.102	35
Ours (MonteBoxFinder)	**0.201**	**0.982**	**0.322**	**0.018**	**37**

Table 6.2 Comparison between our method and our baselines. Our method outperforms all baselines on all metrics computed on ScanNet. We retrieve a more accurate fit, while being able to find more non-intersecting cuboids.

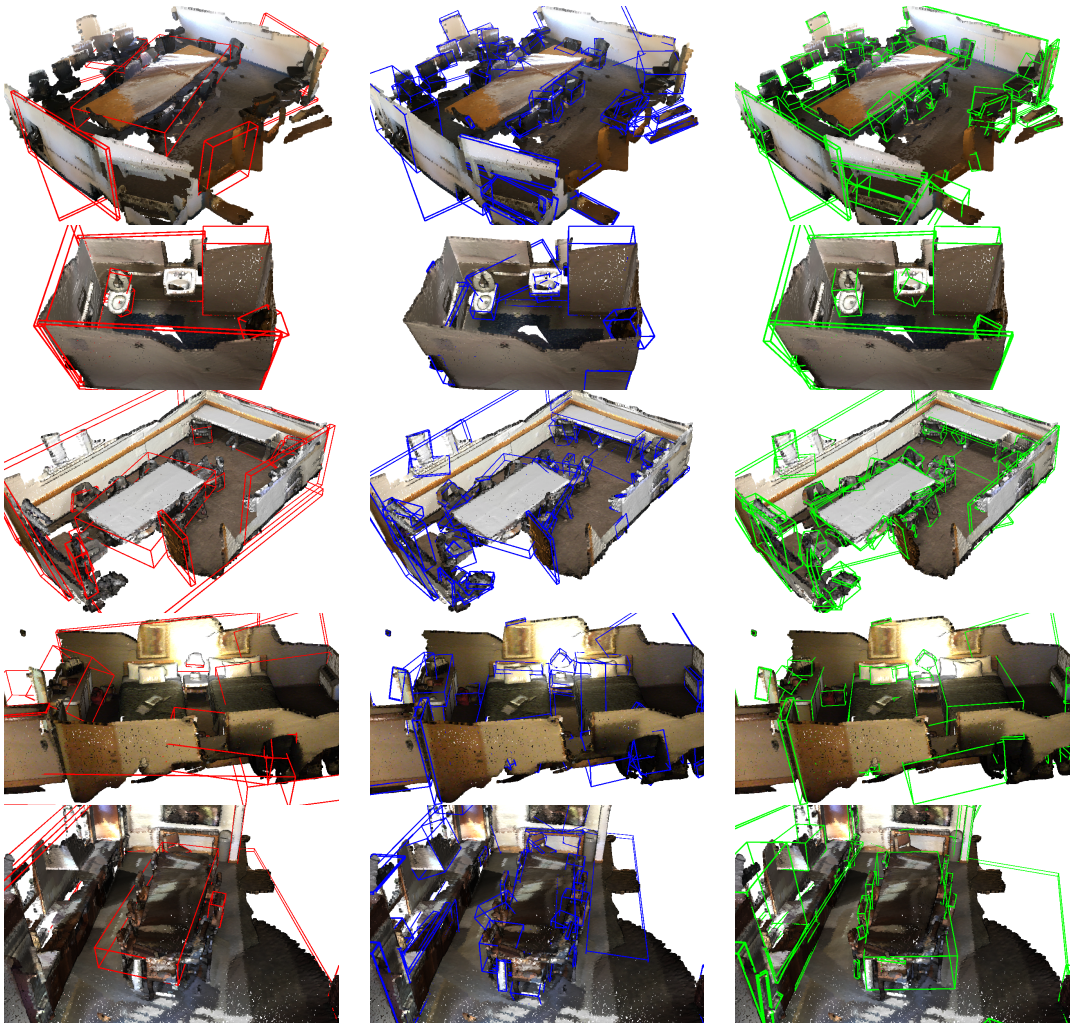
loss, it gets stuck to local minimum solutions consisting of large proposals. It can also provide a complete solution only once it converged, while MCTS, MCTS-Binary and our method can provide a good solution much faster. The table also shows that our algorithm converges significantly faster than MCTS and MCTS-Binary, which was the desired goal. Interestingly, MCTS-Binary performs better than the original MCTS method of Hampali et al. (2021). In the supplementary material, we discuss in details the links between our method and MCTS-Binary.

6.4.5 Qualitative Results

Figure 6.5 shows qualitative results. Hill-climbing focuses on large cuboids to describe the scene. MCTS often selects many true positives but misses some of the proposals because it cannot explore deeper levels of the tree for the given iteration budget. In contrast, our algorithm is able to successfully retrieve cuboid primitives for objects of different sizes, such as walls, floors, and furniture.

6.5 Conclusion

We proposed a method for efficiently and robustly finding a set of cuboids that fits well a 3D point cloud, even under noise and missing data. Our algorithm is not restricted to cuboids, and could consider other primitives. Only a procedure to identify the primitives is required, even if it generates many false positives as our algorithm can reject them. Moreover, the output of our algorithm could be used to generate labeled data for training a deep architecture for fast inference. This could be done to predict cuboids from point clouds, but also from RGB-D images, since the 3D scans of ScanNet were created from RGB-D images By simply reprojecting the cuboids retrieved by our method, we can obtain



Hill-climbing MCTS-Binary MonteBoxFinder (ours)

Fig. 6.5 Qualitative results. Hill-climbing often selects large cuboids that span across multiple different objects (first, third, fourth rows, and fifth rows). MCTS does better, but does not sufficiently explore the solution space (second row). In contrast, our algorithm outperforms both methods and is able to successfully reconstruct many chairs in first, third, and fifth rows, and bedroom furniture in fourth row. *More qualitative results are provided in the supplementary material.*

RGB-D images annotated with the visible cuboids. This concludes our contributions to 3D scene reconstruction from images, which we will discuss in the next and final chapter.
Acknowledgments We would like to thank Pierre-Alain Langlois for his suggestions and help with CGAL. We thank Gul Varol, Van Nguyen Nguyen and Georgy Ponimatkin for our helpful discussions. This project has received funding from the CHISTERA IPALM project.

Chapter 7

Conclusion

In this chapter, we summarize the contributions presented in the thesis. We then discuss a few research directions opened by this work.

7.1 Summary of contributions

This thesis led to the following contributions in monocular depth estimation:

- Reconstructing 3D scenes from images is a difficult problem. In the case where only a *single* color image can be used to infer depth, many challenges arise. Most modern methods therefore employ deep learning to learn to predict good depth maps despite the ill-posedness of the task. We saw in Chapter 3 that most monocular depth estimation methods that focus on overall accuracy lack precision around depth edges. We therefore introduced a geometric constraints at train time in a multi-task learning setup that enforce consistency between surface normals, depth, and occlusion boundaries. This yielded significant improvement in sharpness and accuracy of depth edges, with minimal accuracy loss. We also introduced manual test annotations of occlusion boundaries for the popular NYUv2 dataset, enabling simultaneous evaluation of depth and occlusion boundaries accuracies.

- In Chapter 4, we proposed a new method that enables prediction of sharp depth discontinuities using a new class of networks. Classical fully convolutional networks lack ability to produce strong discontinuities in regression tasks. With the help of Spatial Transformer Networks (Jaderberg et al. (2015)) we designed a depth refinement method that is able to sharpen depth maps around depth edges for all state-of-the-art monocular depth estimation works we considered, without reducing their global accuracy.

- In Chapter 5, we seeked to improve an often overlooked aspect of depth estimation with neural networks: efficiency. Our wavelet-based method exploits the structure of natural depth maps, which are typically piece-wise smooth, with strong discontinuities around occlusion boundaries. Since it is end-to-end differentiable, our method can learn to perform depth estimation through wavelet coefficients estimation, without requiring direct supervision over their values. We finally showed that through a simple tunable parameter, one can choose to trade-off efficiency for accuracy, and still remain competitive in the high efficiency regime.

As a first step to explore primitive decompositions for 3D reconstruction from a single image, in Chapter 6 we developed:

- a method that reconstructs a scene as an arrangement of 3D cuboids from a noisy point cloud obtained by scanning a scene with an RGB-D camera
- a simple modification of the Monte Carlo Tree Search algorithm applied to scene recomposition (Hampali et al. (2021)) to enable efficient and accurate reconstruction as an arrangement of cuboids, given a very large pool of cuboid proposals.

7.2 Future works

In this thesis, we first developed methods to improve the quality of depth maps around occlusion boundaries. This was performed through usage of synthetic data as well as geometric constraints in Chapter 3, and using displacement fields in Chapter 4. While yielding compelling results in depth edge quality, both these works trained neural networks to solve a regression task. However, dense *classification* tasks such as semantic segmentation or foreground/background segmentation often yield crisp, and accurate boundaries. Previous work such as DORN by Fu et al. (2018) exploited this paradigm to improve depth edges, however they produce over-discretized depth maps. An alternative version of our displacement fields could then be to locally classify foreground and background around depth edges in order to impose strong depth gradients.

In Chapter 5, we introduced an efficient method for monocular depth estimation which leverages the sparsity of wavelets when representing piece-wise smooth signals, such as depth maps. One limitation of this work is that this efficiency is theoretical as it is measured in FLOPs and does not translate in direct improvement in speed. In practice, deep learning frameworks such as Pytorch are optimized for convolutions on GPUs. Sparse convolutions on the other hand, cannot benefit from such optimizations that happen

only when convolutions are computed densely. Translating the theoretical efficiency of **WaveletMonodepth** into speed improvements would require more optimization on CPU-like devices, for instance.

Finally in Chapter 6, we explored 3D reconstruction of point clouds using arrangements of cuboids. Our efficient optimization algorithm, implemented as a variant of MCTS by Hampali et al. (2021) provides compelling arrangements after convergence. The loss function used in the optimization problem is global, as it is computed as a variant of Chamfer Distance. In order to improve the proposed solutions even further, one option could be to perform local refinements of the cuboids, drawing inspiration from Iterative Closest Point (ICP), for example. As mentioned in Chapter 6, this work aimed to generate ground truth data for single-image cuboid decomposition for 3D reconstruction. Future work could try to train using this data, after applying the necessary pre-processing steps, such as solving per-view visibility of cuboids for example.

Appendix A

Additional results on SharpNet

A.1 Synthetic Dataset

To train our network to predict geometrically consistent normals, depth, and occluding contours, we used the synthetic dataset from PBRS from Zhang et al. (2017). Indeed, perfect consistency between each output is important to ensure high quality depth maps at occluding contours, but also to ensure good generalization from synthetic to real data. We show in Figure. A.1 a sample from the PBRS dataset, as well as our geometric constraints, which enforce the network to predict outputs consistent with each other, both with synthetic and real images.

A.2 Training Details

Here we present some details about our training method. Since we are performing multi-task learning, each task-attached loss is weighted in the global loss term. We found that learning normals and boundaries first brought the best results. We also present the data augmentation strategy we used while training.

A.2.1 Data Augmentation

We used the following standard data augmentation to train both on PBRS and NYUv2-Depth Silberman et al. (2012):

- random scale with scale factor $s \in [0.5, 2]$: the depth map is divided by s and the z coordinate of the normal map is multiplied by s,

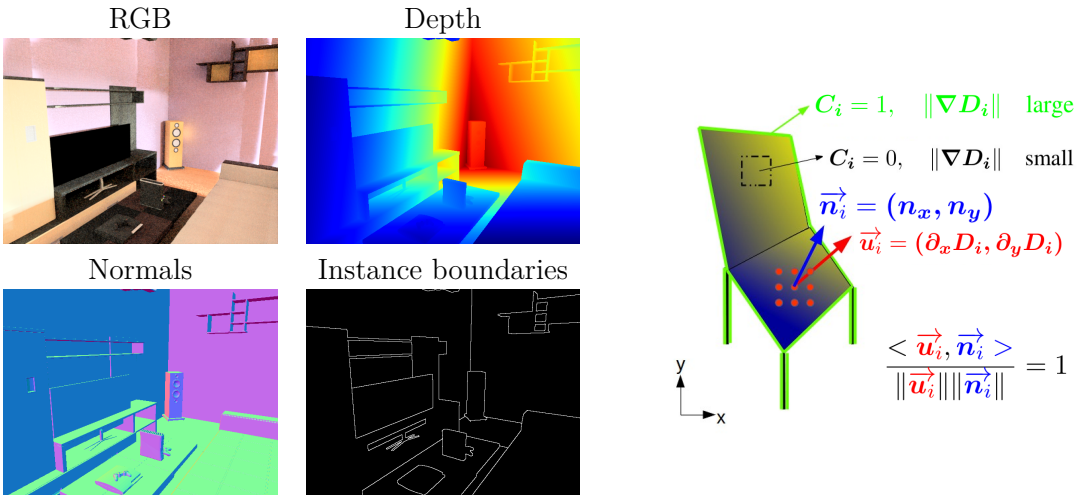


Fig. A.1 A sample of the synthetic PBRS (Zhang et al. (2017)) (left) and geometric constrains on depth, normals and occluding contours (right).

- random rotation of angle $\theta \in [-6°, +6°]$: all corresponding maps are rotated the camera 2D plane and normals maps are recomputed in the camera coordinates (the rotation matrix $R(\theta) \in SO(3)$ is applied on each pixel of rotated normal map)
- random crops of size (320×320),
- random Gamma adjustment using Torchvision `transforms` package, using ratio $\in \left[0.15, \frac{1}{0.15}\right]$

A.2.2 Training Parameters

We recall the loss function equation:

$$\mathcal{L} = \lambda_{\mathrm{d}} \mathcal{L}_{\mathrm{d}}(\boldsymbol{D}, \widehat{\boldsymbol{D}}) + \lambda_{\mathrm{c}} \mathcal{L}_{\mathrm{c}}(\boldsymbol{C}, \widehat{\boldsymbol{C}}) + \lambda_{\mathrm{n}} \mathcal{L}_{\mathrm{n}}(\boldsymbol{N}, \widehat{\boldsymbol{N}}) \ + \mathcal{L}_{\mathrm{dc}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{C}}) + \mathcal{L}_{\mathrm{dn}}(\widehat{\boldsymbol{D}}, \widehat{\boldsymbol{N}}).$$

We detail all training steps in Table. A.1. For all experiments, we used polynomial learning rate decay with power 0.9. We also used weight decay with a decay rate of 1.10^{-6}. At all step, we wait until convergence and pick the model with lowest validation loss. For finetuning on NYUv2-Depth, we freeze normals and occluding contour decoders.

Dataset	batch size	iter size	learning rate	λ_d	λ_c	λ_n	$\mathcal{L}_{dc}$	$\mathcal{L}_{dn}$
PBRS	8	3	1.10^{-2}	1	0.01	20	OFF	OFF
PBRS	8	3	9.10^{-3}	1	0.005	0.5	ON	ON
NYUv2-Depth	8	3	5.10^{-3}	1	0	0	OFF	OFF
NYUv2-Depth	8	3	3.10^{-3}	1	0	0	ON	ON

Table A.1 Training details for each step of our training method. *iter size* stands for the number of batches used per iteration for back-propagation (performed using the average loss computed from each batch loss. λ_c values are this low to rescale the attention loss of Wang et al. (2018a).

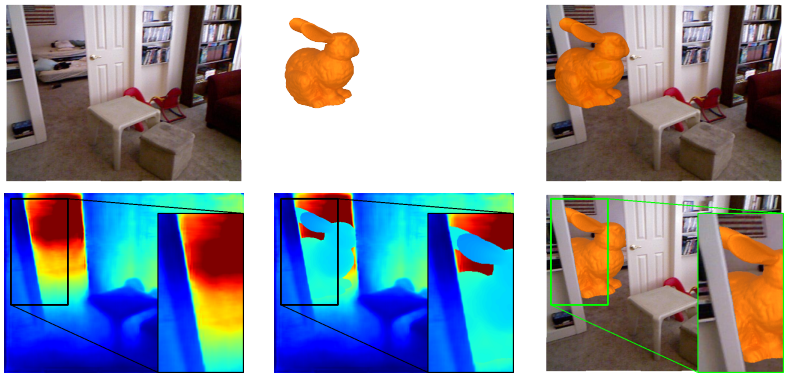


Fig. A.2 Illustration of our occlusion-aware virtual object insertion. Top row (left to right): the original RGB image, the virtual object $\tilde{\boldsymbol{o}}$, an object insertion ignoring occlusion. Bottom row (left to right): our estimated depth map $\boldsymbol{D}$, the augmented depth map when using $\tilde{\boldsymbol{o}}$, the final result.

A.3 Additional Qualitative Results

We show in Figure. A.3 some results of image augmentations: we augment the images by adding a virtual object $\tilde{\boldsymbol{o}}$ in them using both the RGB image and a depth map $\boldsymbol{D}$ of the scene and a depth map $\boldsymbol{D}_{\tilde{\boldsymbol{o}}}$ of the virtual object. This object is inserted with consideration of occlusions, i.e. we only fill pixels such that $\boldsymbol{D}_{\tilde{\boldsymbol{o}}} < \boldsymbol{D}$ with RGB pixels from the rendered object $\tilde{\boldsymbol{o}}$.

Additional results on SharpNet

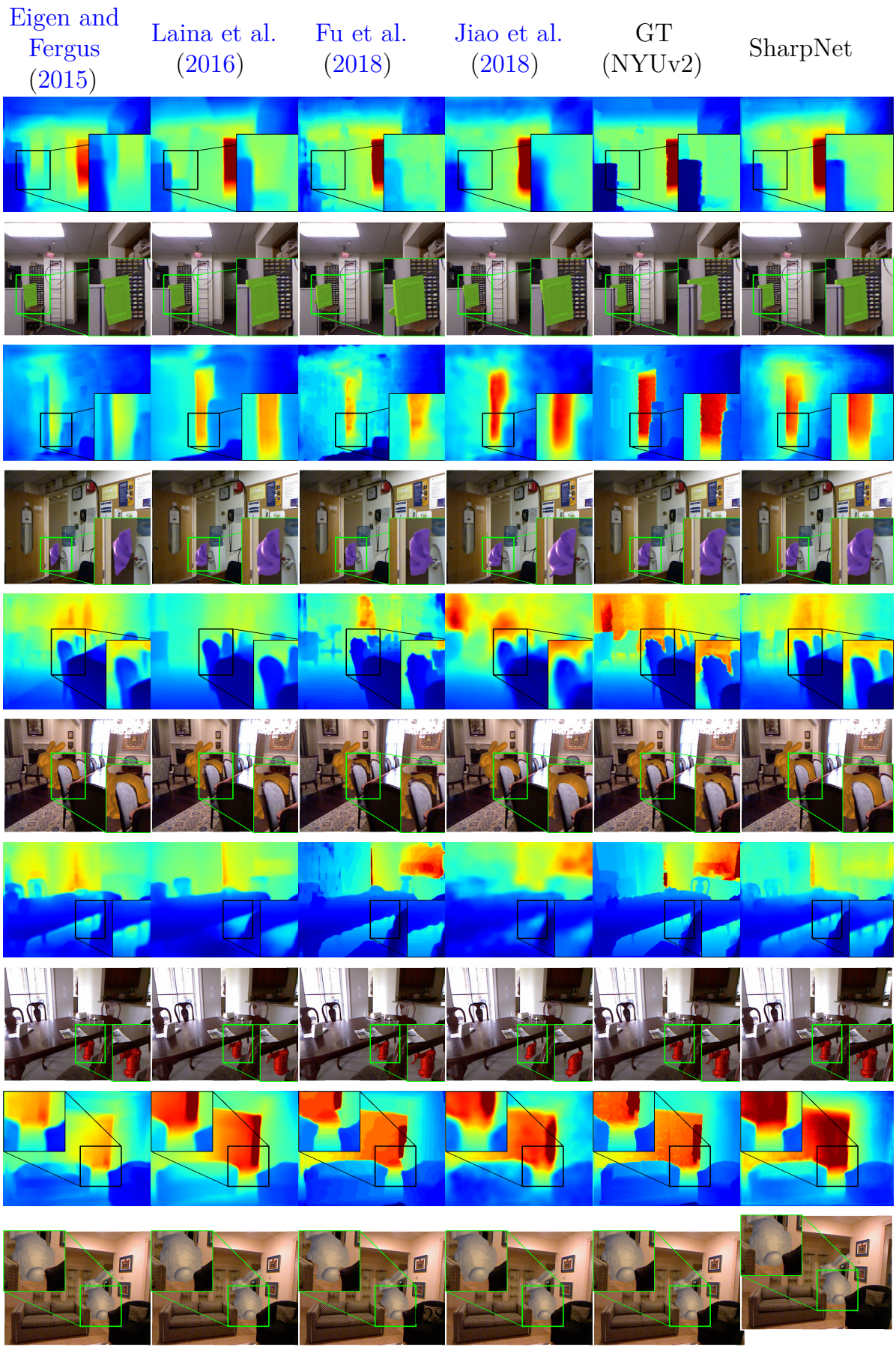


Fig. A.3 More examples of virtual occlusion-aware object insertion using depth maps predicted by different methods as well as along with the Kinect ground truth depth map from the original NYUv2-Depth dataset

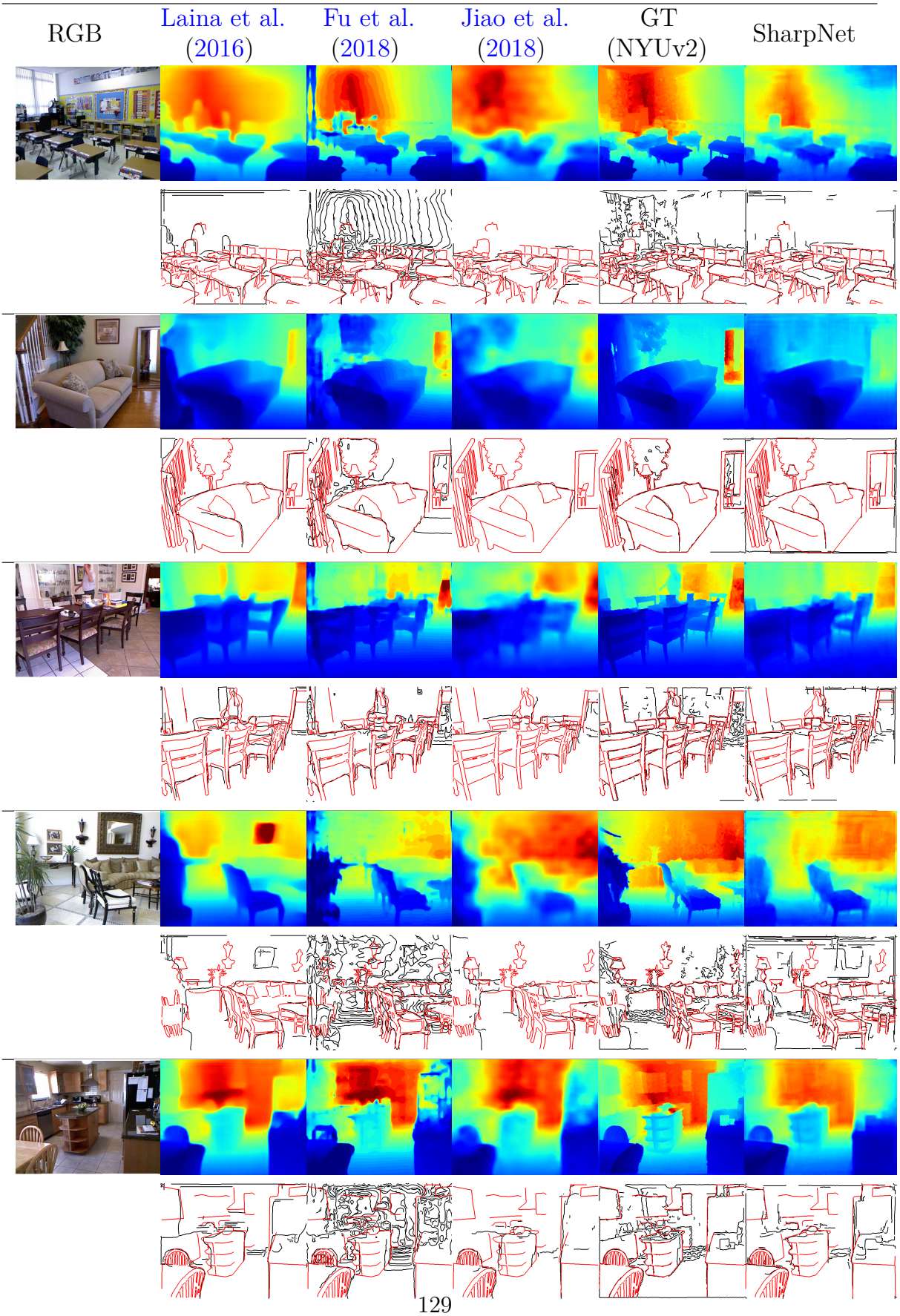


Fig. A.4 More examples of images from our NYUv2-OC dataset and their associated depth map estimate for different methods. Edges (in black) were detected using a Canny edge detector with $\sigma^- = 0.03$ and $\sigma^+ = 0.05$. Our manually annotated ground truth is represented in red.

Appendix B

Additional results on Displacement Fields

B.1 Architecture Details

In Fig. B.1, we detail the architecture of our network, which is composed of two encoders, one for Depth and an optional one for Guidance. We use a single decoder which combines the respective outputs of the Depth and Guidance decoders using residual blocks and skip connections. We give full details of each block in the following.

B.1.1 Depth Encoder

Our Depth is a standard encoder with a cascade of four down-convolutions, denoted *D-DownConv*. The D-DownConv blocks are composed of a convolution layer with 3x3 kernel, followed by a 2x2 MaxPooling, and a LeakyReLU (Maas et al. (2013)) activation. The D-DownConv block convolution layers have respectively [32, 64, 128, 256] channels. They all use batch-normalization, are initialized using Xavier Glorot et al. (2011b) initialization and a Leaky ReLU activation.

B.1.2 Guidance Encoder

Our Guidance encoder is composed of a cascade four of down-convolutions as in He et al. (2016b), which we denote G-DownConv. It is identical to the D-DownConv block described in B.1.1 except that it uses simple ReLU Glorot et al. (2011b) activations. and batch normalization for the convolution layers. The convolution layers have respectively [32, 64, 128, 256] channels.

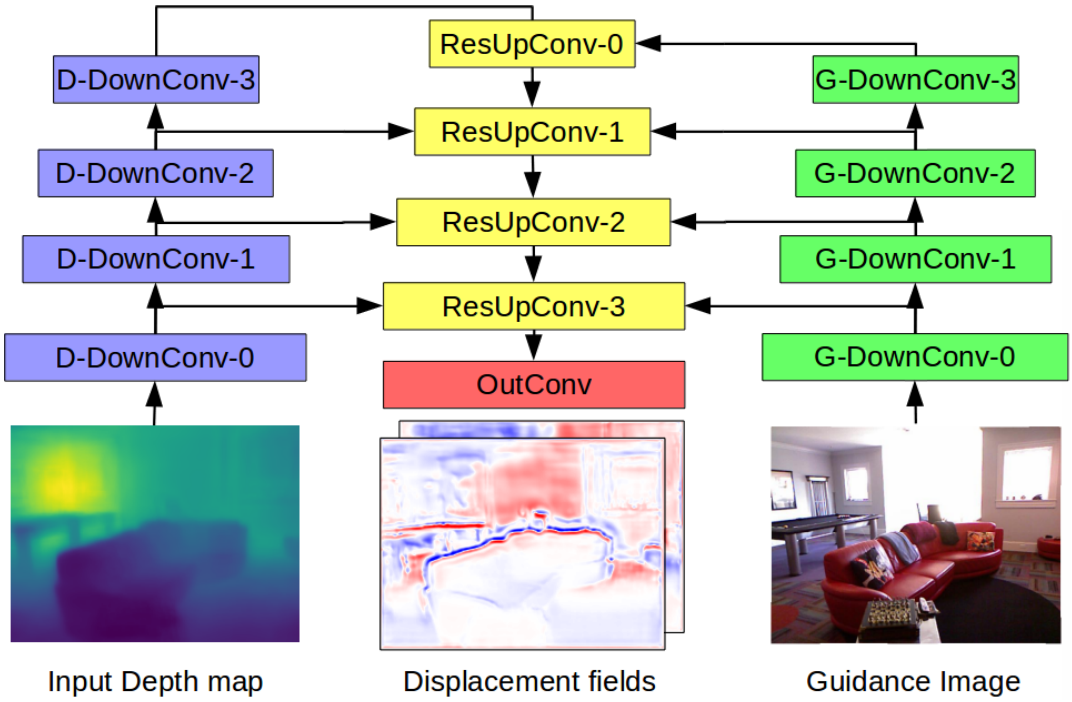


Fig. B.1 Detailed architecture of our displacement field prediction network. Details on the Depth and Guidance encoders are provided in Sections B.1.1 and B.1.2 respectively.

B.1.3 Displacement Field Decoder

The displacement field decoder is composed of a cascade of four ResUpConv blocks detailed in Fig B.1.3.1, and a convolution block OutConv.

B.1.3.1 ResUpConv

The ResUpConv block is the main component of our decoder. It fuses depth and guidance features at multiple scales using skip connections. The block architecture is detailed in Fig B.2. Guidance features are refined using a 3x3 residual convolution layer He et al. (2016b) denoted ResConv3x3. All blocks use batch-normalization, Leaky ReLU Maas et al. (2013) activation and filters weights are initialized using Xavier initialization.

B.1.3.2 Output layers

The final output block *OutConv* is composed of two Conv3x3 layers with batch-normalization and ReLU Glorot et al. (2011b) activation, followed by a simple 3x3 convolution layer without batch-normalization nor activation. The number of channels of those layers are respectively 32, 16, and 2. Weights are initialized using Xavier initialization.

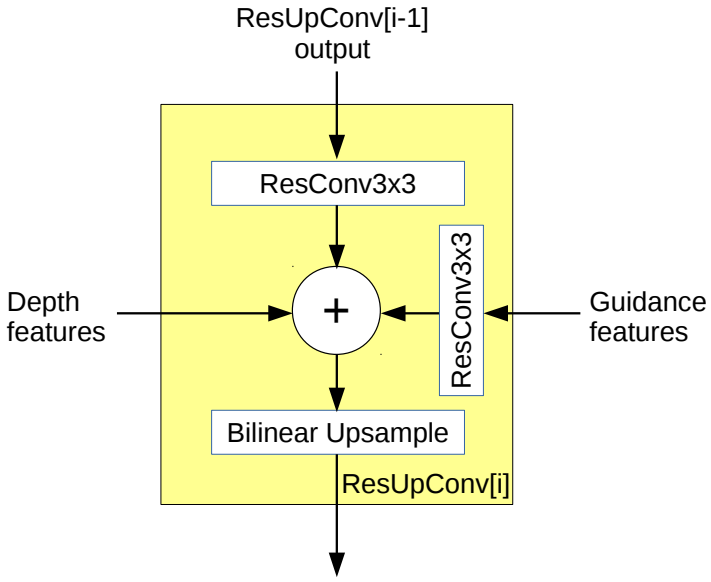


Fig. B.2 Details of the ResUpConv block used in our displacement field decoder.

B.2 Qualitative Results

B.2.1 Comparative Results on 2D Toy Problem

In Fig. B.3, we show qualitative results on the 2D Toy Problem described in Section 3.3 of the main paper. One can see that residual update introduces severe artifacts around edges, producing large and spread out errors around them. Constrastingly, our proposed displacement update recovers sharp edges without degrading the rest of the image. To ensure fair comparison between residual and displacement update methods, identical CNN architectures were used for this experiment except for the last layer which predicts a 2-channel output for the displacement field instead of the 1D-channel output residual.

B.2.2 Comparative Results on NYUv2 Using Different MDE Methods

In Fig B.4, we show qualitative results of our proposed refinement method on different MDE methods (Eigen et al. (2014); Fu et al. (2018); Jiao et al. (2018); Laina et al. (2016); Ramamonjisoa and Lepetit (2019); Yin et al. (2019)) evaluated in the main paper. Our method always improves the sharpness of initial depth map prediction, without degrading the global depth reconstruction.

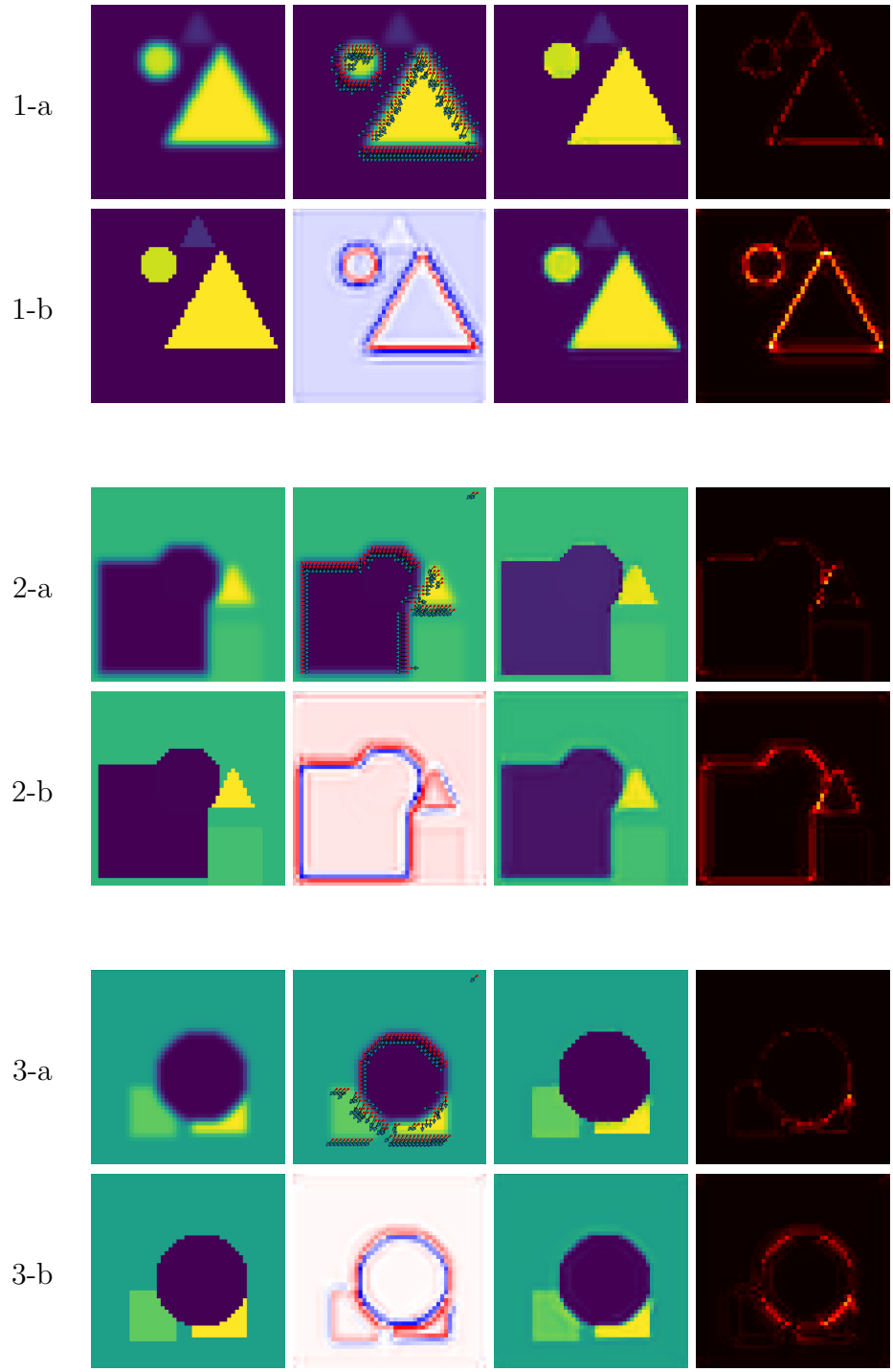


Fig. B.3 Comparison between residual and displacement learning on a toy image sharpening problem for three examples. A blurred input image $\tilde{I}$ is fed through a Convolutional Neural Network which learns to reconstruct the original clean image I. Lines (1,2,3)-a show from left to right: the input image, samples of the dense predicted displacement field, refinement result with displacement update, error map. Lines (1,2,3)-b show, from left to right: the ground truth image, the predicted residual (blue is negative, red is positive, white is zero), refinement result with residual update, error map. While introducing artifacts, residual learning also results in a spread out error map around edges.

Fig. B.4 Refinement results using our method (best seen in color). Each example is represented on two rows, first row being the original predicted depth and second row being the refined depth. First column shows the RGB input images and associated ground truth depth from NYUv2 Silberman et al. (2012). Following columns are refinement results for different methods we evaluated.

B.3 More NYUv2-OC++ Samples

In Fig. B.5 we show several examples of our manually annotated 654 images NYUv2-OC++ dataset, which extends Ramamonjisoa and Lepetit (2019). This dataset is based on the official test split of the popular NYUv2-Depth (Silberman et al. (2012)) depth estimation benchmark.

Fig. B.5 Samples taken from our fine-grained manually annotated NYUv2-OC++, which add occlusion boundaries to the popular NYUv2-Depth (Silberman et al. (2012)) benchmark. We annotated the full official 654 images test set of NYUv2-Depth.

Appendix C

Additional results on WaveletMonodepth

C.1 Network Architectures and Losses

C.1.1 On direct supervision of wavelet coefficients

The previous work WaveletStereo (Yang et al. (2020)) supervises its wavelet based stereo matching method with ground truth wavelet coefficients at the different levels of the decomposition. However, wavelets can only reliably be supervised when ground truth depth -or disparity- is provided and when it does not contain missing values or high-frequency noise, as they show on the synthetic SceneFlow (Mayer et al. (2016)) dataset. The sparsity of ground truth data in the KITTI dataset especially around edges makes it impossible to estimate reliably ground truth wavelet coefficients. On NYUv2, the noise in depth maps is also an issue for direct supervision of wavelets, e.g. with creases in the layout or inaccurate depth edges. This noise also prohibits the use of Semi Global Matching ground truth for wavelet coefficient supervision.

As we show in our work, supervising the network on wavelet reconstructions allows us to ignore missing values and be robust to noisy labels.

C.1.2 Experiments on KITTI

Architecture The architecture we use for our experiments is a modification of the architecture used in Godard et al. (2019), as described in the main paper. In Table C.1, we set out our decoder architecture in detail.

*Work done during an internship at Niantic

Self-supervised losses Our self-supervised losses are as described in Godard et al. (2019), which we repeat here for completeness. Given a stereo pair of images (I_L, I_R), we train our network to predict a depth map D_L, pixel-aligned with the left image. We also assume access to the camera intrinsics K, and the relative camera transformation between the images in the stereo pair $T_{R\to L}$. We use the network's current estimate of depth to synthesise an image $I_{R\to L}$, computed as

$$I_{R\to L} = I_R\Big\langle \text{proj}(D_L, T_{R\to L}, K)\Big\rangle, \tag{C.1}$$

where proj() are the 2D pixel coordinates obtained by projecting the depths D_L into image I_R, and $\big\langle\big\rangle$ is the sampling operator. We follow standard practice in training the model under a photometric reconstruction error pe, so our loss becomes

$$L_p = pe(I_L, I_{R\to L}). \tag{C.2}$$

Following Chen et al. (2019c); Godard et al. (2019) etc. we use a weighted sum of SSIM and L1 losses

$$pe(I_a, I_b) = \alpha\frac{1 - \text{SSIM}(I_a, I_b)}{2} + (1-\alpha)\|I_a - I_b\|_1,$$

where $\alpha = 0.85$. We additionally follow Godard et al. (2019) in using the smoothness loss:

$$L_s = |\partial_x d_L^*|\, e^{-|\partial_x I_L|} + |\partial_y d_L^*|\, e^{-|\partial_y I_L|}, \tag{C.3}$$

where $d_L^* = d_L/\overline{d_L}$ is the mean-normalized inverse depth for image I_L.

When we train on monocular and stereo sequences ('MS'), we again follow Godard et al. (2019) — see our main paper for an overview, and Godard et al. (2019) for full details.

Depth Hints loss When we train with depth hints, we use the proxy loss from Watson et al. (2019), which we recap here. For stereo training pairs, we compute a proxy depth map $\tilde{D}_L$ using semi-global matching Hirschmuller (2005), an off-the-shelf stereo matching algorithm. We use this to create a second synthesized image

$$\tilde{I}_{R\to L} = I_R\Big\langle \text{proj}(\tilde{D}_L, T_{R\to L}, K)\Big\rangle, \tag{C.4}$$

We decide whether or not to apply a supervised loss using $\tilde{D}_L$ as ground truth on a *per-pixel* basis. We only add this supervised loss for pixels where $pe(I_L, \tilde{I}_{R\to L})$ is lower $pe(I_L, I_{R\to L})$. The supervised loss term we use is $\log L_1$, following Watson et al. (2019). For experiments where Depth Hints are used for training, we disable the smoothness loss term.

Additional experiments We additionally tried training using edge-aware sparsity constraints that penalize non-zero coefficients at non-edge regions, by replacing depth gradients with wavelets coefficients in Monodepth's Godard et al. (2017) *disparity smoothness loss*, which unfortunately made training unstable. We also tried to supervise wavelet coefficients using distillation (Aleotti et al. (2021); Hinton et al. (2015)) from a teacher depth network, which resulted in lower performances.

C.1.3 Experiments on NYUv2

Architecture We adapted our architecture from the PyTorch implementation of DenseDepth (Alhashim and Wonka (2018)). Our implementation uses a DenseNet161 encoder instead of a DenseNet169, and a standard decoder with up-convolutions. We first design a baseline that does not use wavelets, using the architecture detailed in Table C.3. Our wavelet adaptation of that baseline is then detailed in Table C.4. For experiments reported in the main paper, we follow the DenseDepth strategy and predict outputs at half the input resolution. Hence, the last level of the depth decoder in Table C.4 is discarded. For experiments using a light-weight decoder discussed later in Section C.4.4, which predicts 224×224 depth maps given a 224×224 input image, we keep all four levels of wavelet decomposition.

Supervised losses For our NYU results in the main paper, we supervise depth using an L1 loss and SSIM:

$$L_D(y, y^*) = \lambda_1 \ell_1(y, y^*) \tag{C.5}$$

where y and y^* are respectively predicted and ground truth depth and $\lambda_1 = 0.1$. Similar to Ramamonjisoa et al. (2020); Ramamonjisoa and Lepetit (2019), we clamp depth between 0.4 and 10 meters.

C.2 Scores on Improved KITTI Ground Truth

We report results on the improved KITTI ground truth (Uhrig et al. (2017)) in Table C.5. As we saw in the main paper, our method is competitive on scores with non-wavelets baselines, but as we have shown our wavelet decomposition enables more efficient predictions.

C.3 Qualitative Results

In this section, we show qualitative results of our method.

In Figures C.3-C.4-C.5 and Figures C.6-C.7-C.8 we first show our sparse prediction process with corresponding sparse wavelets and masks, on the NYUv2 and KITTI datasets respectively. While we only need to compute wavelet coefficients in less than 10% of pixel locations in the decoding process, we show that our wavelets efficiently retain relevant information. Furthermore, we show that wavelets efficiently detect depth edges and their orientation. Therefore, future work could make efficient use of our wavelet based depth estimation method for occlusion boundary detection.

In Figure C.1, we show comparative results between our baseline Depth Hints Watson et al. (2019) and our wavelet based method.

C.4 Exploring Other Efficiency Tracks

Our paper mainly explores computation reduction in the decoder of a UNet-like architecture. However, this direction is orthogonal and complementary with all other complexity reduction lines of research.

Our approach is for example complementary with the FastDepth approach, which consists in reducing the overall complexity of a depth estimation network by compressing it in many dimensions such as (1) the encoder, (2) the decoder (3) the input resolution. They argue that the deep network introduced by Laina *et al.* Laina et al. (2016) suffers from high complexity, while it could largely be reduced. Here we present a set of experiments we conducted to explore these different aspects of complexity reduction.

C.4.1 Experiment with a light-weight MobileNetv2 encoder

First, we replace the costly ResNet (He et al. (2016b)) or DenseNet (Huang et al. (2019, 2017a)) backbone encoders with the efficient MobileNetv2 (Sandler et al. (2018)). Indeed, in contrast with FastDepth, in the main paper, we report results using large encoder

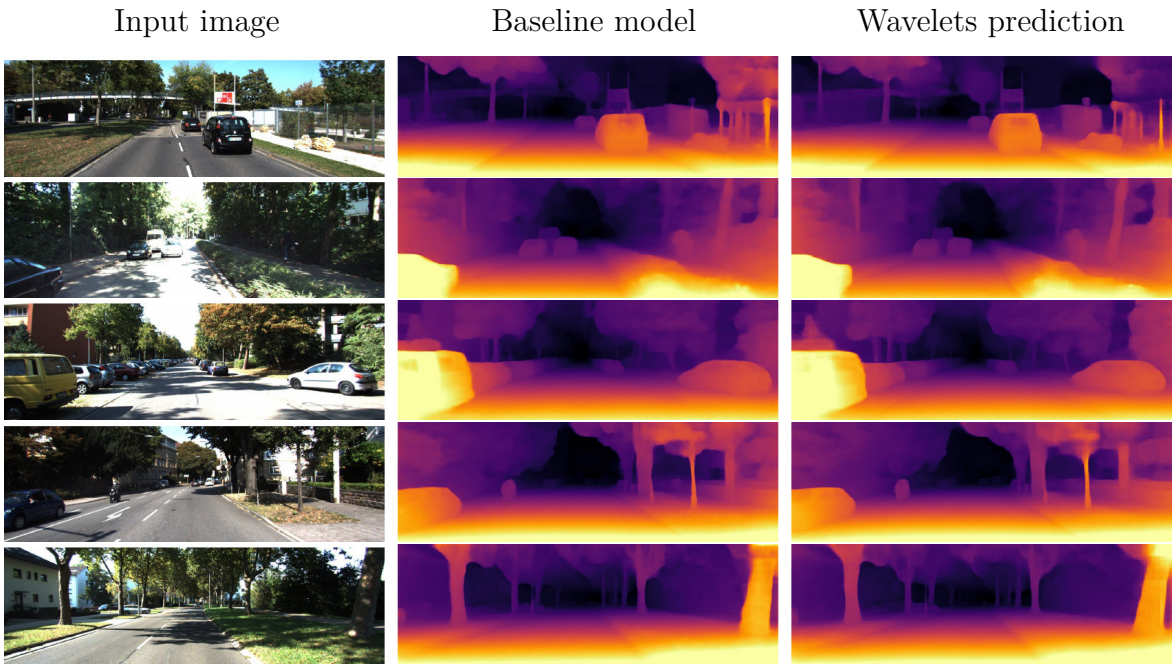


Fig. C.1 Comparing wavelet predictions to a baseline model on the KITTI dataset. On the left we show the input image, and in the middle column we show the prediction from an off-the-shelf Depth Hints ResNet 50 model Watson et al. (2019). On the right we show an equivalently trained ResNet 50 model, but with our wavelets in the decoder. We see that our predictions retain the high quality of the baseline predictions, but are more efficient to predict.

models (Resnet18/50 or Densenet161). Although this helps achieving better scores, we show in Table C.6 and Table C.7 that we can reach close to state-of-the-art results even with a small encoder such as MobileNetv2.

C.4.2 Separable convolutions

Secondly, FastDepth also shows that separable convolutions in their "NNConv" decoder provides the best score-efficiency trade-off. Since this approach is orthogonal to our sparsification method, it therefore complements our method and can be used to improve efficiency. Interestingly, we show in Table C.7 that replacing sparse convolutions with sparse-depthwise separable convolutions works on par with standard convolutions. This can be explained by the fact that **IDWT** is also a separable operation, and therefore efficiently combines with depthwise separable convolutions.

C.4.3 Channel pruning

A popular approach to complexity and memory footprint reduction is channel pruning, which aims at removing some of the unnecessary channel in convolutional layers. Note that our wavelet enabled sparse convolutions are complementary with channel pruning, as can be seen in Figure C.2. While channel pruning can, in practice, greatly reduce both complexity and memory footprint, it requires heavy hyper-parameter search that we therefore choose to leave for future work.

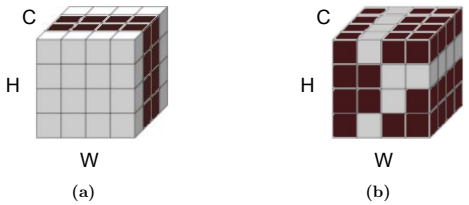


Fig. C.2 Channel pruning (a) vs our sparse computation (b). Our sparse computation enabled by wavelets is complimentary with the channel pruning strategy to reduce the amount of computation, as both computation reduction methods operate in orthogonal dimensions.

C.4.4 Input resolution

Finally, one important factor that makes FastDepth efficient is that it is trained with 224×224 inputs, against our 640×480 input. While our method is best designed for higher-resolution regime where sparsity of wavelets is stronger, we still show that our method achieves decent results even at low-resolution, and report our scores in Table C.8.

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	256	32	econv5	ELU Clevert et al. (2015)
Level 3 coefficients predictions						
iconv4	3	1	256	16	↑upconv5, econv4	ELU
disp4	3	1	1	16	iconv4	Sigmoid
wave4	3	1	3	16	iconv4	Sigmoid
upconv4	3	1	128	16	iconv4	ELU
$\mathbf{IDWT}_3$	-	-	1	8	**disp4**, **wave4**	-
Level 2 coefficients predictions						
iconv3	3	1	128	8	↑upconv4, econv3	ELU
wave3	3	1	3	8	iconv3	Sigmoid
upconv3	3	1	64	8	iconv3	ELU
$\mathbf{IDWT}_2$	-	-	1	8	$\mathbf{IDWT}_3$, **wave3**	-
Level 1 coefficients predictions						
iconv2	3	1	64	4	↑upconv3, econv2	ELU
wave2	3	1	3	4	iconv2	Sigmoid
upconv2	3	1	32	4	iconv2	ELU
$\mathbf{IDWT}_1$	-	-	1	8	$\mathbf{IDWT}_2$, **wave2**	-
Level 0 coefficients predictions						
iconv1	3	1	32	2	↑upconv2, econv1	ELU
wave1	3	1	3	2	iconv1	Sigmoid
$\mathbf{IDWT}_0$	-	-	-	1	$\mathbf{IDWT}_1$, **wave1**	-

Table C.1 Our decoder network architecture for experiments on the KITTI Geiger et al. (2012) dataset using ResNet backbone Here **k** is the kernel size, **s** the stride, **chns** the number of output channels for each layer, **res** is the downscaling factor for each layer relative to the input image, and **input** corresponds to the input of each layer where ↑ is a 2× nearest-neighbor upsampling of the layer. **disp4** is used produce the low-resolution estimate LL_3, while **waveJ** is used to decode $\{\mathsf{LH}_\mathbf{J}, \mathsf{HL}_\mathbf{J}, \mathsf{HH}_\mathbf{J}\}$ at level **J**. **disp4** and **waveJ** are convolution blocks detailed in Table C.2.

disp4 Layer						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
disp4(1)	1	1	**chns**(iconv5) / 4	16	iconv5	LeakyReLU(0.1) Xu et al. (2015)
disp4(2)	3	1	1	16	disp4-1	Sigmoid

Wavelet Decoding Layer - waveJ						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
wave**J**(1+)	1	1	**chns**(iconv[**J**+1])	2^J	iconv[**J**+1]	LeakyReLU(0.1)
wave**J**(2+)	3	1	3	2^J	wave**J**(1+)	Sigmoid
wave**J**(1-)	1	1	**chns**(iconv[**J**+1])	2^J	iconv[**J**+1]	LeakyReLU(0.1)
wave**J**(2-)	3	1	3	2^J	wave**J**(1-)	Sigmoid
substract	1	1	3	2^J	wave**J**(2+), wave**J**(2-)	Linear

Table C.2 Architecture of our wavelet decoding layer used for KITTI experiments J denotes the level of the decoder. **disp4** is used produce the low-resolution estimate LL_3, while **waveJ** is used to decode $\{\mathsf{LH}_\mathbf{J}, \mathsf{HL}_\mathbf{J}, \mathsf{HH}_\mathbf{J}\}$.

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	1104	32	econv5	Linear
iconv4	3	1	552	16	↑upconv5, econv4	LeakyReLU(0.2)
iconv3	3	1	276	8	↑iconv4, econv3	LeakyReLU(0.2)
iconv2	3	1	138	4	↑iconv3, econv2	LeakyReLU(0.2)
iconv1	3	1	69	2	↑iconv2, econv1	LeakyReLU(0.2)
outconv0	1	1	1	2	iconv1	Linear

Table C.3 Architecture of our DenseNet baseline decoder for experiments on the NYUv2 Silberman et al. (2012) dataset Note that as in DenseDepth Alhashim and Wonka (2018) we produce a depth map at half-resolution. Table adapted from Godard et al. (2019).

Depth Decoder						
layer	**k**	**s**	**chns**	**res**	**input**	**activation**
upconv5	3	1	1104	32	econv5	Linear
Level 3 coefficients predictions						
iconv4	3	1	552	16	↑upconv5, econv4	LeakyReLU(0.2)
disp4	1	1	1	16	upconv5	Linear
wave4	3	1	3	16	upconv5	Linear
$\mathbf{IDWT}_3$	-	-	1	8	disp4, wave4	-
Level 2 coefficients predictions						
iconv3	3	1	276	8	↑iconv4, econv3	LeakyReLU(0.2)
wave3	3	1	3	8	iconv3	Linear
$\mathbf{IDWT}_2$	-	-	1	4	$\mathbf{IDWT}_3$, wave3	-
Level 1 coefficients predictions						
iconv2	3	1	138	4	↑iconv2, econv2	LeakyReLU(0.2)
wave2	3	1	3	4	iconv2	Linear
$\mathbf{IDWT}_1$	-	-	1	2	$\mathbf{IDWT}_2$, wave2	-

Table C.4 Our decoder network architecture for experiments on the NYUv2 (Silberman et al. (2012)) dataset Note that since like in DenseDepth (Alhashim and Wonka (2018)) we produce a depth map at half-resolution, we only need to predict wavelet coefficients until quarter-resolution. Table adapted from Godard et al. (2019).

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
Godard et al. (2019)	Monodepth2 Resnet18	✓	S	192 × 640	0.079	0.512	3.721	0.131	0.924	0.982	0.994
	WaveletMonodepth Resnet18	✓	S	192 × 640	0.084	0.523	3.807	0.137	0.914	0.980	0.994
	WaveletMonodepth Resnet50	✓	S	192 × 640	0.081	0.477	3.658	0.133	0.920	0.981	0.994
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	192 × 640	0.085	0.487	3.670	0.131	0.917	0.983	0.996
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.083	0.476	3.635	0.129	0.920	0.983	0.995
	Depth Hints Resnet50	✓	S_{SGM}	192 × 640	0.081	0.432	3.510	0.124	0.924	0.985	0.996
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.081	0.449	3.509	0.125	0.923	0.986	0.996
Godard et al. (2019)	Monodepth2 Resnet18	✓	MS	192 × 640	0.084	0.494	3.739	0.132	0.918	0.983	0.995
	WaveletMonodepth Resnet18	✓	MS	192 × 640	0.085	0.497	3.804	0.134	0.912	0.982	0.995
Watson et al. (2019)	Depth Hints	✓	MS + S_{SGM}	192 × 640	0.087	0.526	3.776	0.133	0.915	0.982	0.995
	WaveletMonodepth Resnet18	✓	MS + S_{SGM}	192 × 640	0.086	0.497	3.699	0.131	0.914	0.983	0.996
Godard et al. (2019)	Monodepth2 Resnet18	✓	S	320 × 1024	0.082	0.497	3.637	0.132	0.924	0.982	0.994
	WaveletMonodepth Resnet18	✓	S	320 × 1024	0.080	0.443	3.544	0.130	0.919	0.983	0.995
	WaveletMonodepth Resnet50	✓	S	320 × 1024	0.076	0.413	3.434	0.126	0.926	0.984	0.995
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	320 × 1024	0.077	0.404	3.345	0.119	0.930	0.988	0.997
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.078	0.397	3.316	0.121	0.928	0.987	0.997
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.074**	0.363	3.198	**0.114**	**0.936**	**0.989**	**0.997**
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	**0.074**	**0.357**	**3.170**	**0.114**	**0.936**	**0.989**	**0.997**

Table C.5 Quantitative results on the improved KITTI benchmark. We compare our method to our baselines on the KITTI (Geiger et al. (2012)) improved dataset introduced by Uhrig et al. (2017), using the Eigen split. *Data column* (data source used for training): S is for self-supervised training on stereo images, MS is for models trained with both M (forward and backward frames) and S data and S_{SGM} refers to the extra stereo ground truth which was used in Watson et al. (2019).

Cit.	Method	PP	Data	H × W	Abs Rel	Sq Rel	RMSE	RMSE log	δ<1.25	δ<1.25^2	δ<1.25^3
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	192 × 640	0.106	0.780	4.695	0.193	0.875	0.958	0.980
	WaveletMonodepth MobileNetv2	✓	S_{SGM}	192 × 640	0.109	0.851	4.754	0.194	0.870	0.957	0.980
	WaveletMonodepth Resnet18	✓	S_{SGM}	192 × 640	0.107	0.829	4.693	0.193	0.873	0.957	0.980
	WaveletMonodepth Resnet50	✓	S_{SGM}	192 × 640	0.105	0.813	4.625	0.191	0.879	0.959	0.981
Watson et al. (2019)	Depth Hints	✓	S_{SGM}	320 × 1024	0.099	0.723	4.445	0.187	0.886	0.961	0.982
	WaveletMonodepth MobileNetv2	✓	S_{SGM}	320 × 1024	0.104	0.772	4.545	0.188	0.880	0.960	0.982
	WaveletMonodepth Resnet18	✓	S_{SGM}	320 × 1024	0.102	0.739	4.452	0.188	0.883	0.960	0.981
	Depth Hints Resnet50	✓	S_{SGM}	320 × 1024	**0.096**	**0.710**	4.393	0.185	0.890	**0.962**	0.981
	WaveletMonodepth Resnet50	✓	S_{SGM}	320 × 1024	0.097	0.718	**4.387**	**0.184**	**0.891**	**0.962**	**0.982**

Table C.6 Quantitative results on the KITTI dataset using MobileNetv2 encoder. We evaluate results of our method using a lighter encoder on KITTI (Geiger et al. (2012)), using the Eigen split. *Data column* (data source used for training): S is for self-supervised training on stereo images, MS is for models trained with both M (forward and backward frames) and S data and S_{SGM} refers to the extra stereo ground truth which was used in Watson et al. (2019).

Method	Encoder	Depthwise	H × W	Abs Rel	RMSE	log_{10}	δ_1	δ_2	δ_3	ϵ_{acc}	ϵ_{comp}
Dense baseline	DenseNet161	-	480 × 640	0.1277	**0.5479**	**0.0539**	0.8430	**0.9681**	**0.9917**	**1.7170**	**7.0638**
Ours	DenseNet161	-	480 × 640	**0.1258**	0.5515	0.0542	**0.8451**	**0.9681**	**0.9917**	1.8070	7.1073
Ours	DenseNet161	✓	480 × 640	0.1275	0.5771	0.0557	0.8364	0.9635	0.9897	2.0133	7.1903
Dense baseline	MobileNetv2	-	480 × 640	0.1772	0.6638	0.0731	0.7419	0.9341	0.9835	1.8911	7.7960
Ours	MobileNetv2	-	480 × 640	0.1727	0.6776	0.0732	0.7380	0.9362	0.9844	1.9732	7.9004
Ours	MobileNetv2	✓	480 × 640	0.1734	0.6700	0.0731	0.7391	0.9347	0.9844	2.3036	8.0538

Table C.7 Quantitative results on NYUv2 (Silberman et al. (2012)) using depth-wise convolutions and light-weight encoder We show that our method is compatible with other efficiency seeking approaches such as depth-wise separable convolutions and lower complexity encoders.

Method	Encoder	Depthwise	H × W	Abs Rel	RMSE	log_{10}	δ_1	δ_2	δ_3
Dense baseline	DenseNet161	-	224 × 224	0.1278	0.5715	0.0557	0.8368	0.9620	0.9901
Ours	DenseNet161	-	224 × 224	0.1279	0.5651	0.0549	0.8399	0.9652	0.9899
Ours	DenseNet161	✓	224 × 224	0.1304	0.5775	0.0564	0.8329	0.9613	0.9892
Dense baseline	MobileNetv2	-	224 × 224	0.1505	0.6221	0.0632	0.7984	0.9526	0.9878
Ours	MobileNetv2	-	224 × 224	0.1530	0.6409	0.0655	0.7844	0.9500	0.9864
Ours	MobileNetv2	✓	224 × 224	0.1491	0.6463	0.0646	0.7880	0.9506	0.9871

Table C.8 Quantitative results on NYUv2(Silberman et al. (2012)) using depth-wise convolutions and light-weight encoder We show that our method is compatible with other efficiency seeking approaches such as depth-wise separable convolutions and lower complexity encoders.

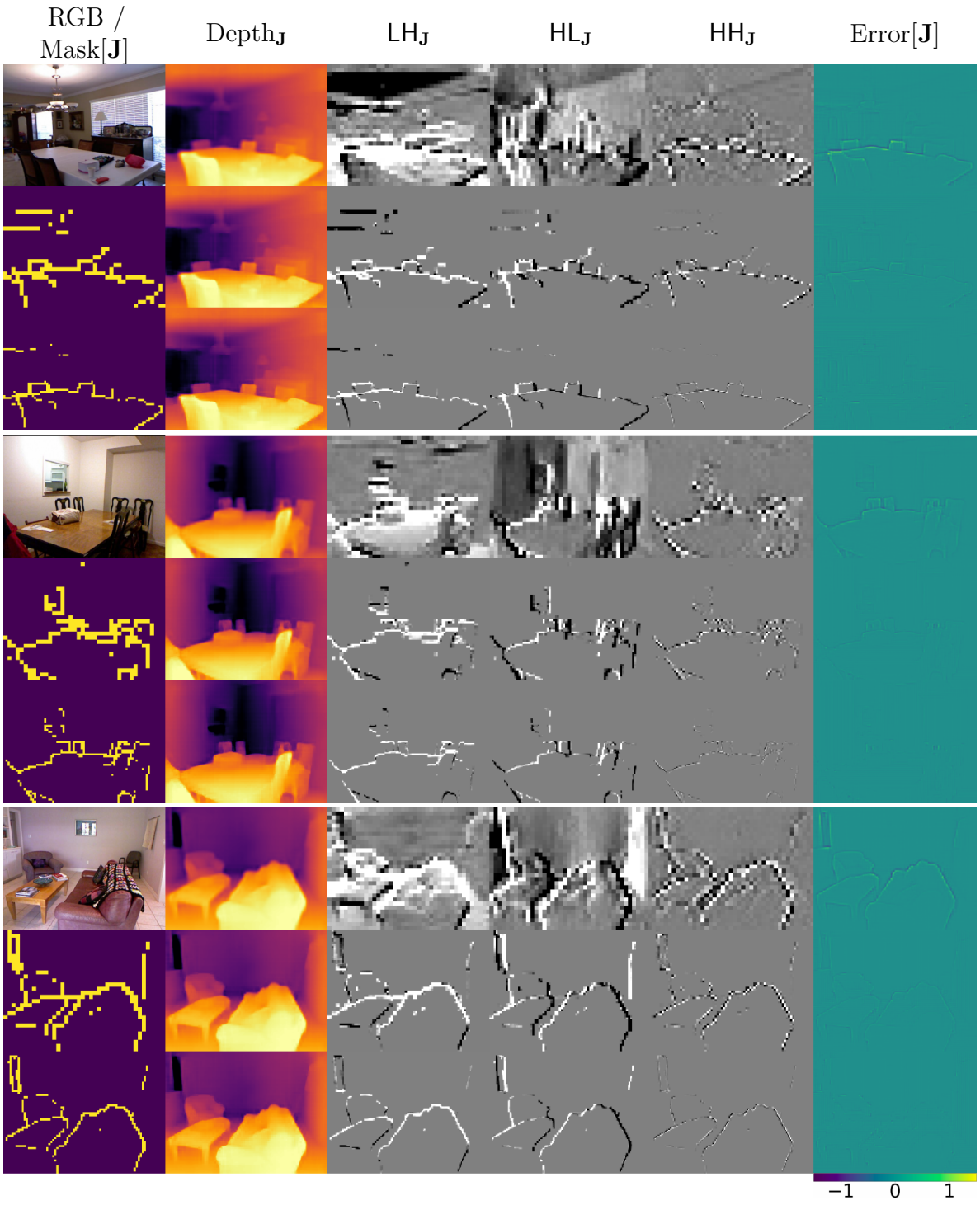


Fig. C.3 Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (1/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale **J** in the decoder from lowest to highest scale (decreasing **J**), as well as the (signed-)error between Depth$_J$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

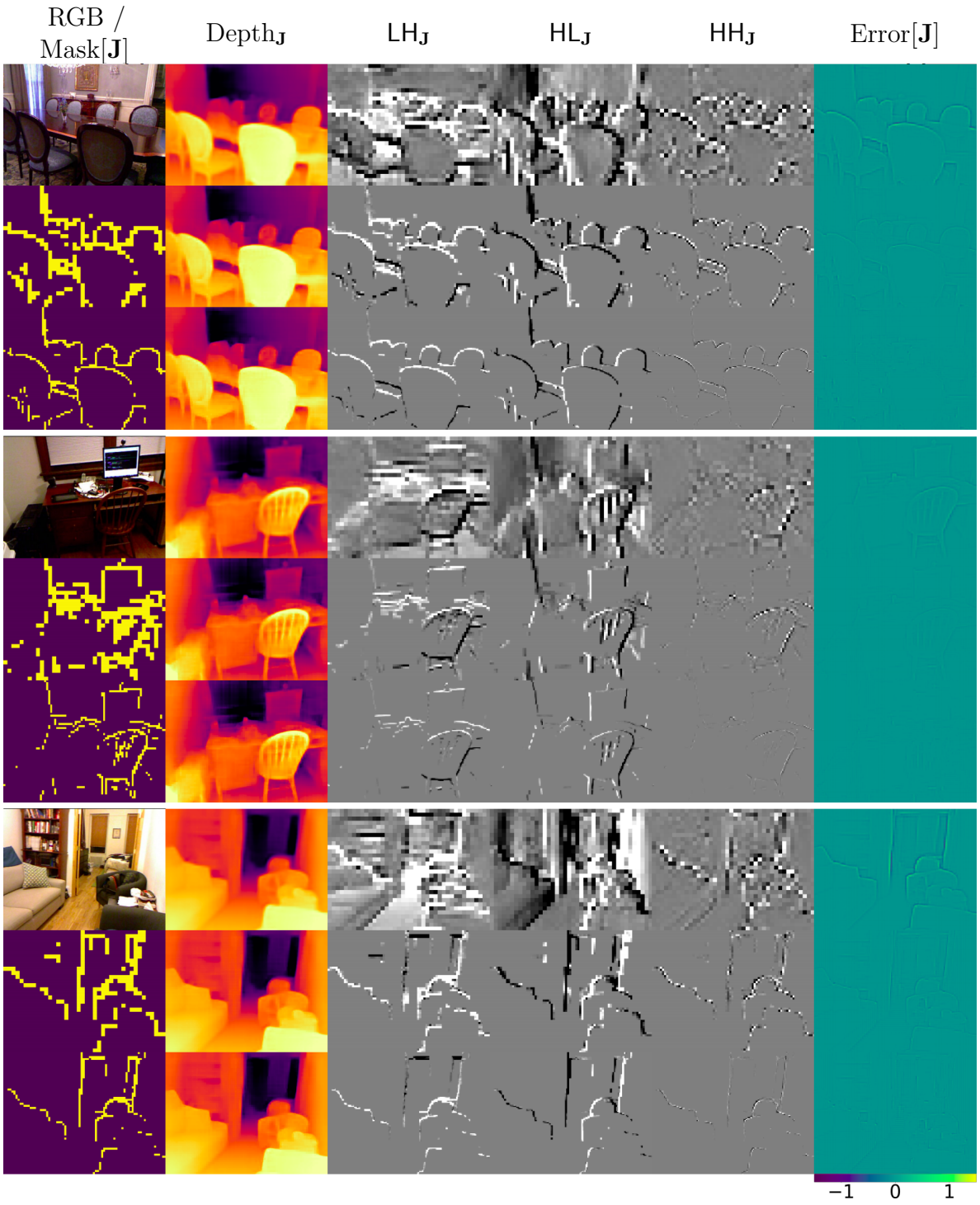


Fig. C.4 Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (2/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale $\mathbf{J}$ in the decoder from lowest to highest scale (decreasing $\mathbf{J}$), as well as the (signed-)error between $\text{Depth}_\mathbf{J}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

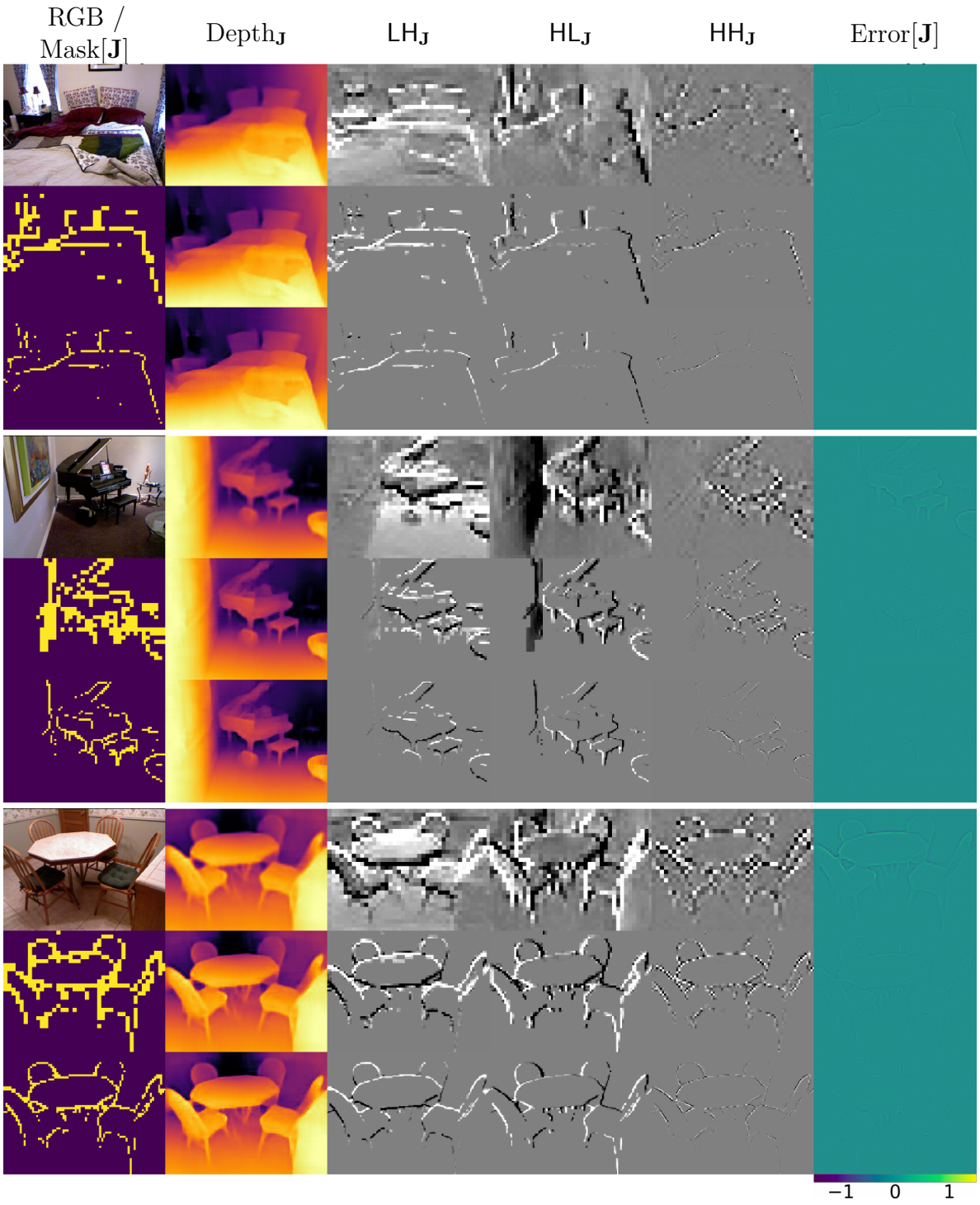


Fig. C.5 Qualitative results of predicted wavelets coefficients of depth maps on the NYU dataset (3/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.04$. For each block of results, each row shows coefficients and depth maps obtained at scale $\mathbf{J}$ in the decoder from lowest to highest scale (decreasing $\mathbf{J}$), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-1.5m, 1.5m]$.

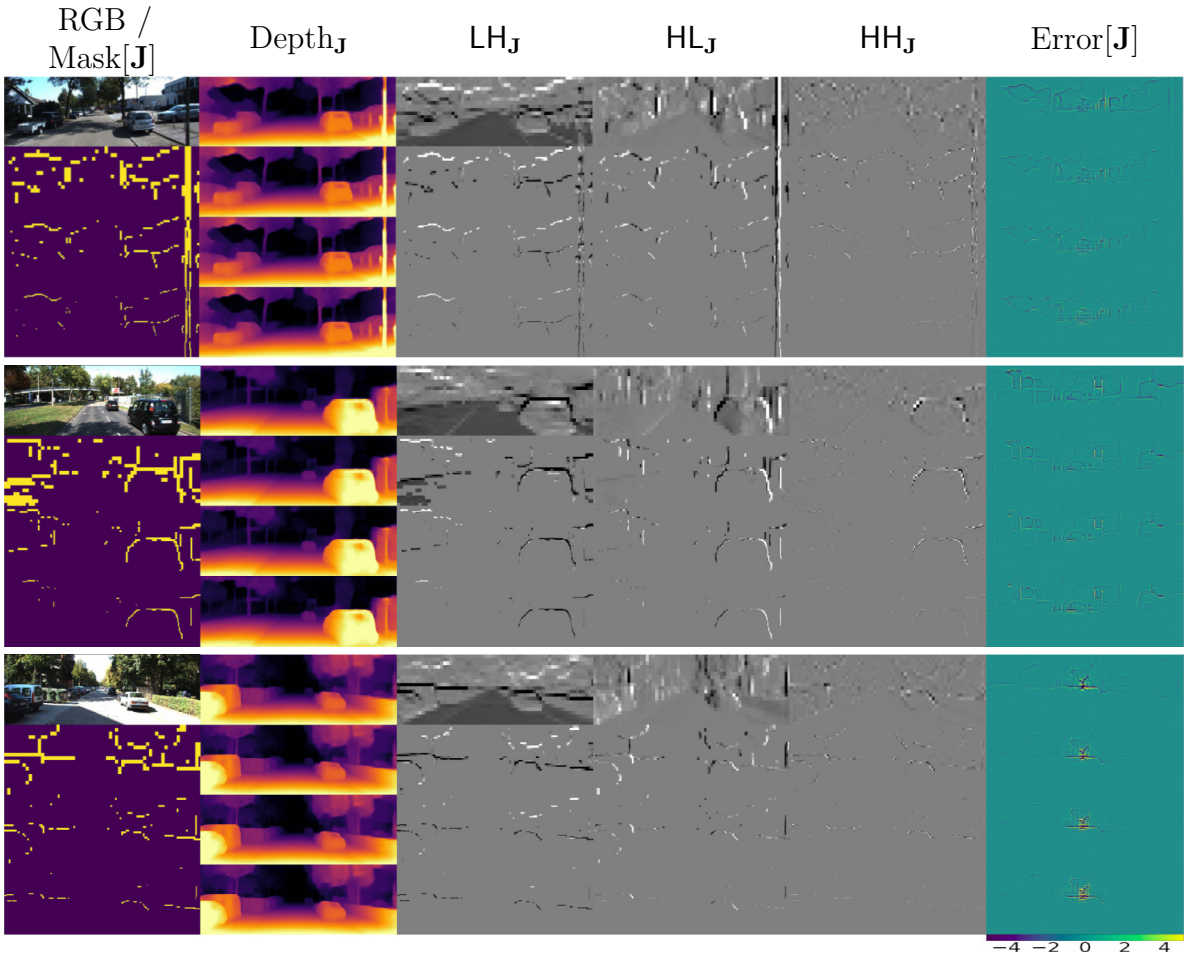


Fig. C.6 Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (1/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale $\mathbf{J}$ in the decoder from lowest to highest scale (decreasing $\mathbf{J}$), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.

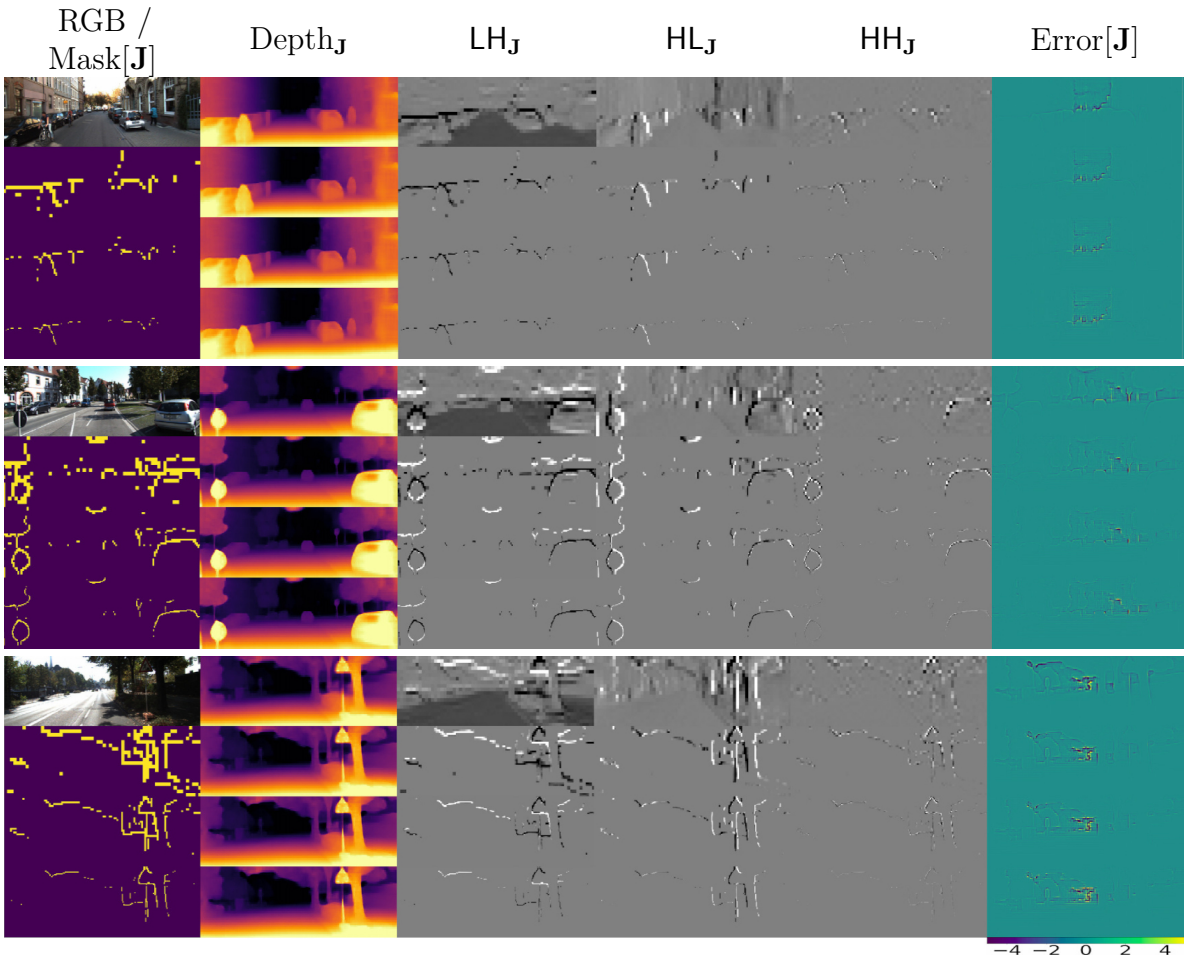


Fig. C.7 Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (2/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale $\mathbf{J}$ in the decoder from lowest to highest scale (decreasing $\mathbf{J}$), as well as the (signed-)error between $\text{Depth}_{\mathbf{J}}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.

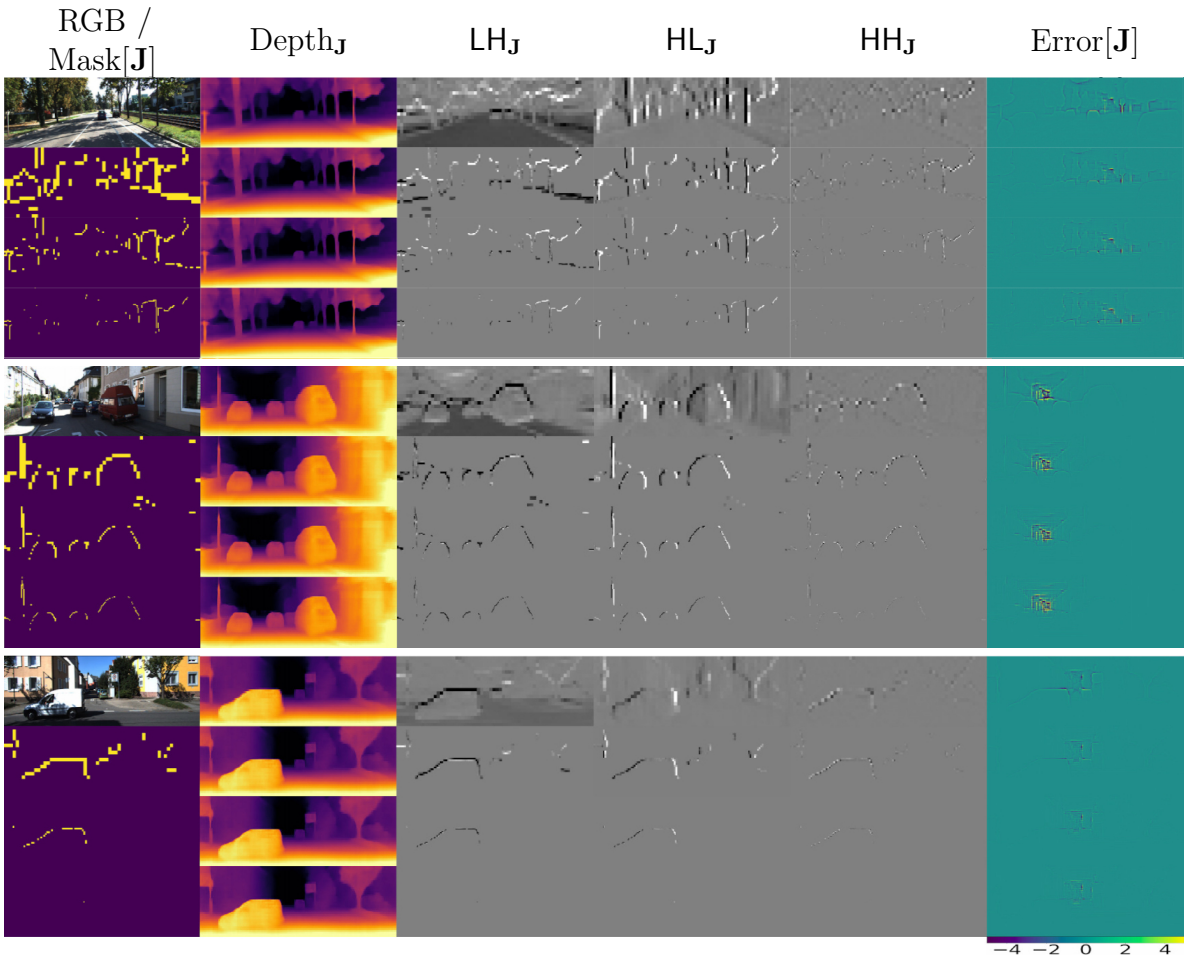


Fig. C.8 Qualitative results of predicted wavelets coefficients of depth maps on the KITTI dataset (3/3). Our predicted wavelets have two desirable properties: they are sparse, allowing for efficient computation, and they are accurately located around depth edges without needing to supervise them. Results are obtained with $\eta = 0.05$. For each block of results, each row shows coefficients and depth maps obtained at scale $\mathbf{J}$ in the decoder from lowest to highest scale (decreasing $\mathbf{J}$), as well as the (signed-)error between Depth$_\mathbf{J}$ and the Depth map obtained with dense wavelet coefficients at all scales. Error is displayed within range $[-5m, 5m]$.

Appendix D

Additional results on MonteBoxFinder

D.1 More on Cuboids

D.1.1 Constructing Cuboids from Pairs of Plane Segments

In this section, we provide more details regarding the construction of cuboids given a pair of plane segments $\pi_A = (X_A, \boldsymbol{N}_A)$ and $\pi_B = (X_B, \boldsymbol{N}_B)$.

D.1.1.1 Checking planes adjacency

We recall that two planes (π_A, π_B) should be considered for constructing a cuboid only if they fulfill two requirements: *alignment* and *proximity*.

Proximity Two plane segments are adjacent if they verify the *proximity* criterion, which requires that they have at least one connected component, such that $\min(\text{ChamferDistance}(X_A \rightarrow X_B), \text{ChamferDistance}(X_B \rightarrow X_A)) < \gamma$, where γ is a small 3D distance.

Alignment Two plane segments $\pi_A = (X_A, \boldsymbol{N}_A)$ and $\pi_B = (X_B, \boldsymbol{N}_B)$ are aligned if they are either "orthogonal enough" or "co-linear enough"; This corresponds to $|\boldsymbol{N}_A^T \boldsymbol{N}_B| < \alpha$ or $|\boldsymbol{N}_A^T \boldsymbol{N}_B| > \beta$, respectively, where $\alpha << 1$ and $\beta \lesssim 1$.
In our experiments we set $(\alpha, \beta, \gamma) = (0.3, 0.7, 0.05m)$.

D.1.1.2 Getting cuboids main axes

We can now construct *two* orthonormal bases $\mathcal{B}_A = (\boldsymbol{u}_A, \boldsymbol{v}_A, \boldsymbol{w}_A)$ and $\mathcal{B}_B = (\boldsymbol{u}_B, \boldsymbol{v}_B, \boldsymbol{w}_B)$ of vectors using Gram-Schmidt orthonormalization with N_A or N_B as the first vector alternatively, as shown in Equations (D.1a) and (D.1b).

$$\begin{aligned}
\boldsymbol{u}_A &:= \frac{\boldsymbol{N}_A}{\|\boldsymbol{N}_A\|_2} \\
\boldsymbol{v}_A &:= \boldsymbol{N}_B - \frac{\boldsymbol{u}_A^T \boldsymbol{N}_B}{\|\boldsymbol{N}_B\|_2} \boldsymbol{u}_A \\
\boldsymbol{v}_A &\leftarrow \frac{\boldsymbol{v}_A}{\|\boldsymbol{v}_A\|_2} \\
\boldsymbol{w}_A &:= \boldsymbol{u}_A \times \boldsymbol{v}_A,
\end{aligned} \tag{D.1a}$$

$$\begin{aligned}
\boldsymbol{u}_B &:= \frac{\boldsymbol{N}_B}{\|\boldsymbol{N}_B\|_2} \\
\boldsymbol{v}_B &:= \boldsymbol{N}_A - \frac{\boldsymbol{u}_B^T \boldsymbol{N}_A}{\|\boldsymbol{N}_A\|_2} \boldsymbol{u}_B \\
\boldsymbol{v}_B &\leftarrow \frac{\boldsymbol{v}_B}{\|\boldsymbol{v}_B\|_2} \\
\boldsymbol{w}_B &:= \boldsymbol{u}_B \times \boldsymbol{v}_B,
\end{aligned} \tag{D.1b}$$

D.1.1.3 Computing the final cuboids

Based on the two cuboids bases, we compute their sizes by simply projecting all 3D points $X \in X_A \cup X_B$ and computing the minimum and maximum of the projections along each axes of $\mathcal{B}_A$ and $\mathcal{B}_B$. This results in two bounding boxes aligned with $\mathcal{B}_A$ and $\mathcal{B}_B$ respectively, which both enclose all points in $X_A \cup X_B$.

D.1.2 Computing Intersections with `isCompatible`

In order to check compatibility between cuboids s_1 and s_2, we design a variation of an Intersection-over-Union criterion, replacing the *Union* with the volume of the smallest cuboid between s_1 and s_2. In order to compute the volume of the intersection, we approximate volumes by sampling points in both s_1 and s_2 and counting points that are inside both. Full details of the procedure used in `isCompatible` are given in Algorithm 5. In practice we use an intersection threshold $\eta = 10\%$.

D.2 More About our Baselines

D.2.1 The Hill-Climbing Algorithm

The Hill-Climbing algorithm Skiena (2010) is a naive greedy descent algorithm that constructs a solution iteratively, where at each iteration, it comprehensively searches for the proposal that best improves the objective function of a solution $\mathcal{S}_F$, while leaving the solution valid *i.e.* with no incompatibilities. If no proposal is available nor can improve

the objective function, the algorithm stops ×. A pseudo-code for the Hill-Climbing algorithm is provided in Algorithm 6.

Sub-optimal solutions In practice, Hill-Climbing leads to sub-optimal solutions where the algorithm gets stuck into a local minimum. This is because the algorithm first greedily fits large regions of the scene, therefore employing large `Cuboid` ; This makes a lot of potentially good cuboids unavailable, as they would intersect with that large `Cuboid` .

Sub-optimality of evaluations Hill-climbing has to evaluate the complete objective function ● *each time* it considers a primitive, which is particularly costly at the beginning of the algorithm where the solution $\mathcal{S}_F$ is still empty, since no `Cuboid` candidate would intersect with it. After selecting a large `Cuboid` , the set of available, *i.e.* compatible cuboids gets dramatically reduced, hence resulting in an acceleration of the search of Hill-Climbing as shown in Figure 6.4, which however converges to sub-optimal solutions. In contrast, MCTS and our algorithm evaluate the objective function only at the end of an iteration when a complete solution is complete.

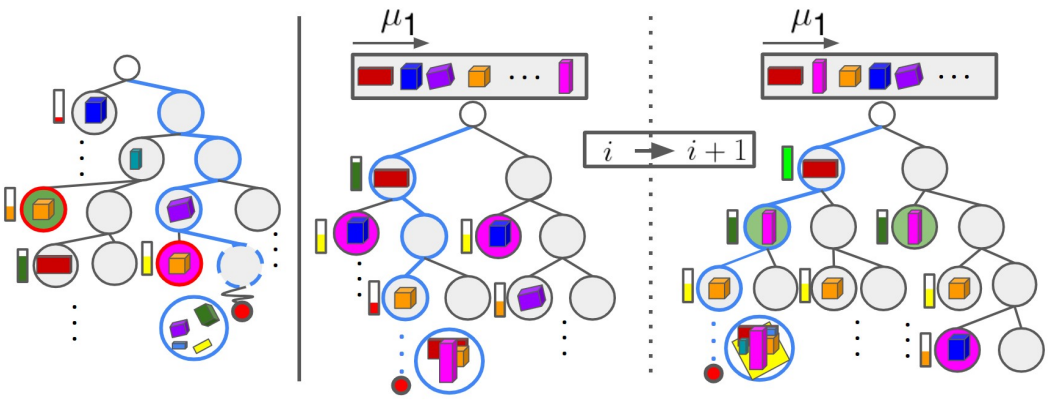
D.2.2 Binary-Tree MCTS

In our non-binary tree adaptation of MCTS, the search algorithm spends many iterations on iterating the first levels of the tree which might contain many, mutually incompatible, primitives. Therefore, this can limit the exploitation capabilities of MCTS as the algorithm prioritizes nodes that have not been visited yet. In our binary-tree adaptation of MCTS, every level of the tree corresponds to selecting or skipping a primitive. The resulting tree structure, hence, trades tree-breadth for tree-depth, which enables better exploitation. However, due to a large depth of the tree, MCTS does not explore solutions in the bottom of the tree, hence we observed only minor improvements over its non-binary adaptation. We argue that our MonteBoxFinder method can be seen as an adaptive version of binary-tree MCTS. In contrast to binary-tree MCTS, as we show in Figure D.1, the tree equivalent of our method is able to adapt its structure during the search and enable better update mechanism leading to faster convergence.

D.3 More results

D.3.1 Qualitative results

We show more qualitative results in Figure D.2.



Binary-tree MCTS 2 iterations of our algorithm interpreted with a binary tree

Fig. D.1 We observe that behavior of our algorithm can be interpreted as an adaptive binary-tree MCTS, even though we do not explicitly define a tree structure. As MCTS is bound by its tree structure, it will invest iterations into exploring primitives in the upper part of the tree, even those with low confidence, visualized as **colored bars**. Further more, as indicated with **colored circles**, MCTS models confidences of same primitives in different parts of the tree independently. Blue circles indicate a selection path of a single MCTS iteration that fails to extract meaningful proposals due to aforementioned difficulties. In contrast, our method sorts at each iteration primitives according to their confidences μ_1 and will focus more easily on more promising primitives. In addition, as indicated by **colored circles** we only model a single confidence per primitive. These features enable our method to converge faster to good solutions in practice.

D.3.2 Progress plots

We show more progress plots in Figure D.3.

Algorithm 5: The `isCompatible` function

procedure `isCompatible`(s_1, $\mathcal{S}$, η)

Result: Returns True if `Cuboid` s_1 is compatible with all `Cuboid` in $\mathcal{S}$

Input : Cuboids `Cuboid` $\mathcal{S}$, threshold η

if ($\forall s_2 \in \mathcal{S}$, `IntersectionOverVolume`($s_1$,$s_2$)$> \eta$) **then**

return False ;

return True ;

procedure `IntersectionOverVolume`(s_1, s_2)

Input : Cuboids s_1 and s_2

Volume of `Cuboid` s_1 V_1 := Volume(s_1)

Volume of `Cuboid` s_1 V_2 := Volume(s_2)

Number of samples $N_{samples}$:= 5000;

Number of samples from s_1 in s_2 $N_{1\subset 2}$:= 0;

Number of samples from s_2 in s_1 $N_{2\subset 1}$:= 0;

`// Sample 3D points within both cuboids` s_1 `and` s_2

$\mathcal{X}_1$:= sample_points_inside(s_1, N) ;

$\mathcal{X}_2$:= sample_points_inside(s_2, N) ;

`// Count points sampled in` s_1 `which are also inside` s_2

for ($x \in \mathcal{X}_1$) {

if $x \in s_2$ **then**

$N_{1\subset 2} \leftarrow N_{1\subset 2} + 1$;

}

`// Count points sampled in` s_2 `which are also inside` s_1

for ($x \in \mathcal{X}_2$) {

if $x \in s_1$ **then**

$N_{2\subset 1} \leftarrow N_{2\subset 1} + 1$;

}

`// Compute approximation of the intersection volume`

$$\text{Intersection} := \frac{V_1 \cdot N_{2\subset 1} + V_2 \cdot N_{1\subset 2}}{2N_{samples}};$$

return Intersection/(min(V_1, V_2)) ;

Algorithm 6: Hill-climbing algorithm

Result: Set of selected `Cuboid` $\mathcal{S}_F$
Input : Set of proposal `Cuboid` $\mathcal{S}$;
Threshold η ;
Final solution $\mathcal{S}_F := \emptyset$;
Current best loss $\ell^* := +\infty$;
Current best `Cuboid` $s^* := \emptyset$;
Set of available `Cuboid` $\mathcal{S}_A := \mathcal{S}$;
Evaluations counter $N_{eval} := 0$;

```
while S_A ≠ ∅ do
    s* ← ∅;
    for ( s ∈ S_A ) {
        if s.isCompatible(s_f, η) then
            S_F.add(s);
            ℓ ← evalObjFunc(S_F);
            N_eval ← N_eval + 1;
            if ℓ < ℓ* then
                ℓ* ← ℓ;
                s* ← s;
            S_F.remove(s);
        else
            S_A.remove(s);
        end
    }
    S_F.add(s*);
end
return S_F, N_eval
```

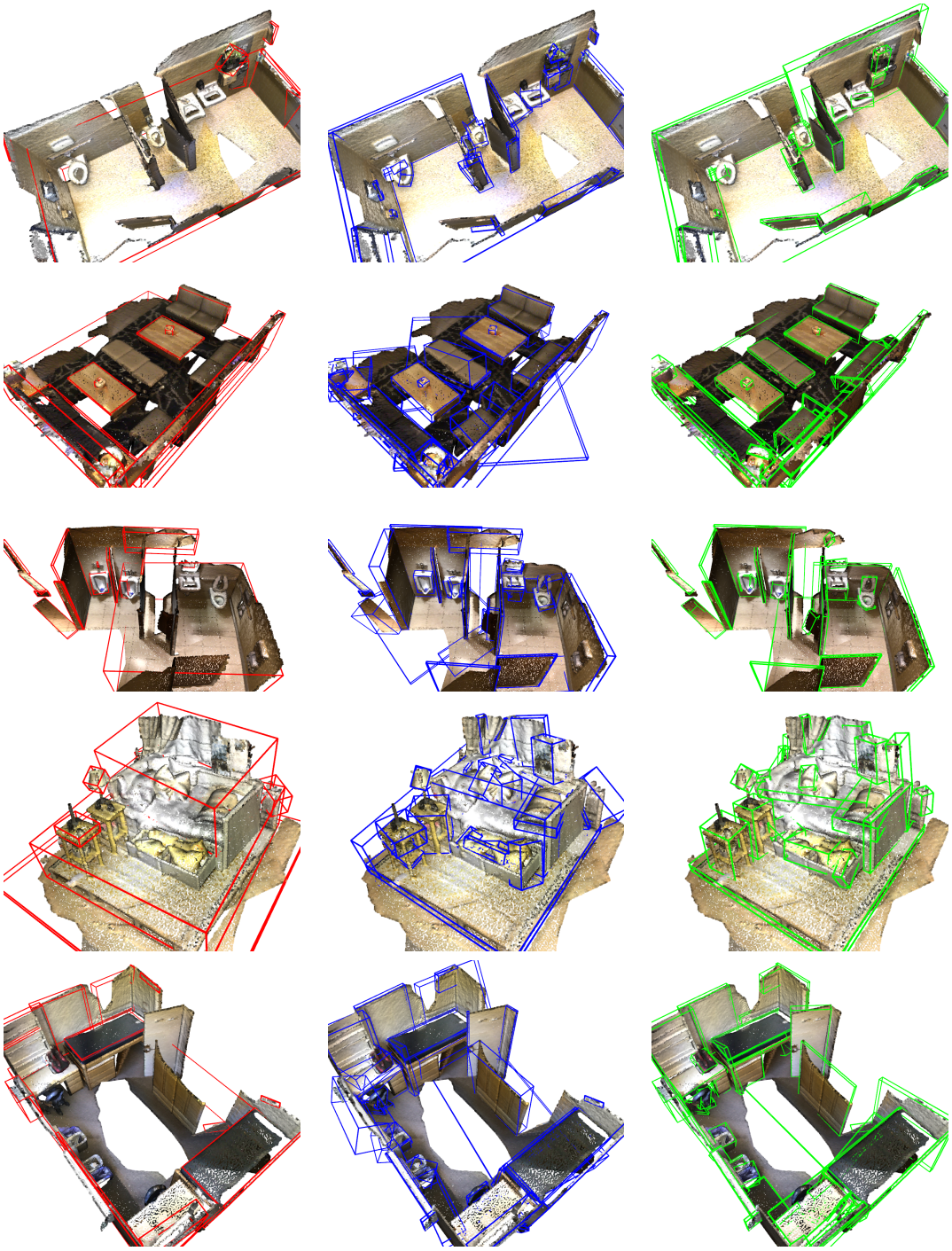


Fig. D.2 Qualitative results. Hill-climbing often selects large cuboids that span across multiple different objects. MCTS does better, but sometimes yields outliers (second and fifth row). In contrast, our algorithm outperforms both methods and is able to successfully reconstruct many more details.

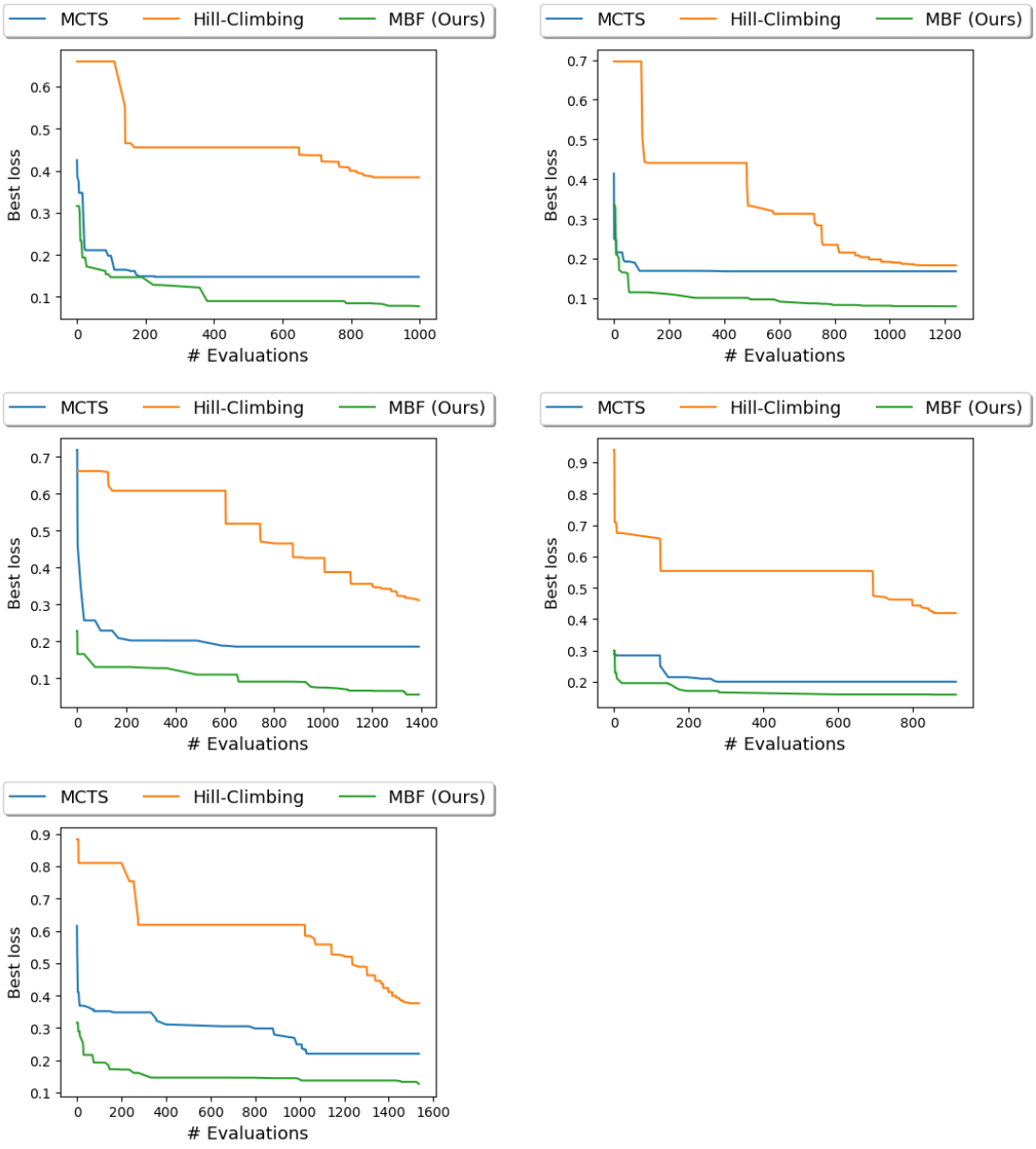


Fig. D.3 Samples of progress plots. Our method consistently outperforms its baselines, i.e. the Hill-Climbing algorithm and MCTS, as it converges faster to a better solution. These plots correspond to the scene examples in Figure D.2

References

Aleotti, F., Tosi, F., Poggi, M., and Mattoccia, S. (2018). Generative adversarial networks for unsupervised monocular depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Aleotti, F., Zaccaroni, G., Bartolomei, L., Poggi, M., Tosi, F., and Mattoccia, S. (2021). Real-time single image depth perception in the wild with handheld devices. *Sensors*.

Alhashim, I. and Wonka, P. (2018). High Quality Monocular Depth Estimation via Transfer Learning. *arXiv:1812.11941*.

Andrieu, C., Freitas, N. D., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*.

Antequera, M. L., Gargallo, P., Hofinger, M., Bulò, S. R., Kuang, Y., and Kontschieder, P. (2020). Mapillary planet-scale depth dataset. In *European Conference on Computer Vision*, pages 589–604. Springer.

Anwar, S. and Barnes, N. (2019). Real image denoising with feature attention. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3155–3164.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.

Barron, J. and Malik, J. (2012a). Color Constancy, Intrinsic Images, and Shape Estimation. In *European Conference on Computer Vision*.

Barron, J. and Malik, J. (2012b). Shape, Albedo, and Illumination from a Single Image of an Unknown Object. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Barron, J. and Poole, B. (2016). The Fast Bilateral Solver. In *European Conference on Computer Vision*.

Barron, J. T. and Malik, J. (2015). Shape, Illumination, and Reflectance from Shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687.

Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., and Kritchman, S. (2020). Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR.

References

Bhat, S. F., Alhashim, I., and Wonka, P. (2021). Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018.

Boulch, A. (2020). Convpoint: Continuous convolutions for point cloud processing. *Computers and Graphics*, 88:24 – 34.

Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez liebana, D., Samothrakis, S., and Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1:1–43.

Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6).

Cao, Y., Zhao, T., Xian, K., Shen, C., and Cao, Z. (2018). Monocular Depth Estimation with Augmented Ordinal Depth Relationships. *IEEE Transactions on Image Processing*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. (2017). Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chang, C.-T., Gorissen, B., and Melchior, S. (2011). Fast Oriented Bounding Box Optimization on the Rotation Group $SO(3, R)$. *ACM Trans. Graph.*, 30(5).

Chen, W., Fu, Z., Yang, D., and Deng, J. (2016). Single-image depth perception in the wild. *Advances in neural information processing systems*, 29.

Chen, X., Chen, X., and Zha, Z.-J. (2019a). Structure-aware residual pyramid network for monocular depth estimation.

Chen, Y., Huang, S., Yuan, T., Qi, S., Zhu, Y., and Zhu, S.-C. (2019b). Holistic++ Scene Understanding: Single-View 3D Holistic Scene Parsing and Human Pose Estimation with Human-Object Interaction and Physical Commonsense. In *International Conference on Computer Vision*.

Chen, Y., Schmid, C., and Sminchisescu, C. (2019c). Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*.

Choi, W., Chao, Y.-W., Pantofaru, C., and Savarese, S. (2013). Understanding Indoor Scenes Using 3D Geometric Phrases. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Cotter, F. (2019). Uses of complex wavelets in deep convolutional neural networks (doctoral thesis). *https://doi.org/10.17863/CAM.53748*, Chapter 3.

CS Kumar, A., Bhandarkar, S. M., and Prasad, M. (2018). Monocular depth prediction using generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 300–308.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017a). ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Dai, A., Nießner, M., Zollhöfer, M., Izadi, S., and Theobalt, C. (2017b). Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1.

Darmon, F., Bascle, B., Devaux, J., Monasse, P., and Aubry, M. (2021). Deep multi-view stereo gone wild. *International Conference on 3D Vision*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Deng, R., Shen, C., Liu, S., Wang, H., and Liu, X. (2018). Learning to Predict Crisp Boundaries. In *European Conference on Computer Vision*.

Deng, X., Yang, R., Xu, M., and Dragotti, P. L. (2019). Wavelet domain style transfer for an effective perception-distortion tradeoff in single image super-resolution. In *ICCV*.

Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., and Aubry, M. (2019). Learning Elementary Structures for 3D Shape Generation and Matching. In *Advances in Neural Information Processing Systems*, pages 7433–7443.

Dollár, P. and Zitnick, C. L. (2013). Structured Forests for Fast Edge Detection. In *International Conference on Computer Vision*.

References

Dollár, P. and Zitnick, C. L. (2015). Fast Edge Detection Using Structured Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570.

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627.

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eigen, D. and Fergus, R. (2015). Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *International Conference on Computer Vision*.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In *Advances in Neural Information Processing Systems*.

Everingham, M., Gool, L. V., Williams, C., Winn, J. M., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal on Computer Vision*, 88(2):303–338.

Fan, H., Su, H., and Guibas, L. J. (2017). A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Favaro, P. and Soatto, S. (2005). A geometric approach to shape from defocus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):406–417.

Fei, X., Wang, A., and Soatto, S. (2018). Geo-Supervised Visual Depth Prediction. *IEEE Robotics and Automation Letters*, 4:1661–1668.

Firman, M. (2016). RGBD Datasets: Past, Present and Future. In *CVPR Workshop on Large Scale 3D Data: Acquisition, Modelling and Analysis*.

Frankot, R. T. and Chellappa, R. (1988). A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on pattern analysis and machine intelligence*, 10(4):439–451.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep Ordinal Regression Network for Monocular Depth Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Fua, P. (1991). Combining Stereo and Monocular Information to Compute Dense Depth Maps That Preserve Depth Discontinuities. In *IJCAI*, pages 1292–1298.

Fukushima, K. (1988). Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition. *Neural Networks*.

Furukawa, Y., Hernández, C., et al. (2015). Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

Geifman, A., Galun, M., Jacobs, D., and Basri, R. (2022). On the spectral bias of convolutional neural tangent and gaussian process kernels. *arXiv preprint arXiv:2203.09255*.

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

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*.

Geiger, D., Ladendorf, B., and Yuille, A. (1995). Occlusions and Binocular Stereo. *International Journal on Computer Vision*, 14:211–226.

Geng, J. (2011). Structured-light 3d surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160.

Glorot, X., Bordes, A., and Bengio, Y. (2011a). Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*.

Glorot, X., Bordes, A., and Bengio, Y. (2011b). Deep Sparse Rectifier Neural Networks. pages 315–323.

Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *ICCV*.

Godard, C., Mac Aodha, O., and J. Brostow, G. (2017). Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *IEEE Conference on Computer Vision and Pattern Recognition*.

GonzalezBello, J. L. and Kim, M. (2020). Forget about the lidar: Self-supervised depth estimators with med probability volumes. *Advances in Neural Information Processing Systems*, 33:12626–12637.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Grabner, A., Roth, P. M., and Lepetit, V. (2019). Gp2c: Geometric projection parameter consensus for joint 3d pose and focal length estimation in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Groenendijk, R., Karaoglu, S., Gevers, T., and Mensink, T. (2020). On the benefit of adversarial training for monocular depth estimation. *Computer Vision and Image Understanding*, 190:102848.

Groueix, T., Fisher, M., Kim, V. G., Russell, B., and Aubry, M. (2018). AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. (2021). Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199.

Guo, T., Mousavi, H. S., Vu, T. H., and Monga, V. (2017). Deep wavelet prediction for image super-resolution. In *CVPR Workshops*.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2020). Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364.

Gupta, A., Efros, A. A., and Hebert, M. (2010). Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *European Conference on Computer Vision*, pages 482–496, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gupta, S., Arbeláez, P., and Malik, J. (2013). Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Gupta, S., Girshick, R., Arbelaez, P., and Malik, J. (2014). Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *European Conference on Computer Vision*.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69:331–371.

Hampali, S., Stekovic, S., Sarkar, S. D., Kumar, C. S., Fraundorfer, F., and Lepetit, V. (2021). Monte Carlo Scene Search for 3D Scene Understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Han, J., Shao, L., Xu, D., and Shotton, J. (2013). Enhanced Computer Vision With Microsoft Kinect Sensor: A Review. *IEEE Transactions on Cybernetics*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017a). Mask R-CNN. In *International Conference on Computer Vision*.

He, K., Sun, J., and Tang, X. (2013). Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1397–1409.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

He, Y., Zhang, X., and Sun, J. (2017b). Channel pruning for accelerating very deep neural networks. In *ICCV*.

Heise, P., Klose, S., Jensen, B., and Knoll, A. (2013). PM-Huber: PatchMatch with Huber Regularization for Stereo Matching. In *IEEE International Conference on Computer Vision*, pages 2360–2367.

Henzler, P., Mitra, N. J., and Ritschel, T. (2019). Escaping plato's cave: 3d shape from adversarial rendering. In *The IEEE International Conference on Computer Vision (ICCV)*.

Heo, M., Lee, J., Kim, K.-R., Kim, H.-U., and Kim, C.-S. (2018). Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement. In *European Conference on Computer Vision*.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv:1503.02531*.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*.

Hirschmuller, H. (2007). Stereo processing by semiglobal matching and mutual information. *PAMI*.

Hoiem, D., Efros, A., and Hebert, M. (2011). Recovering Occlusion Boundaries from an Image. *International Journal on Computer Vision*, 91(3).

Hoiem, D., Efros, A. A., and Hebert, M. (2007). Recovering Surface Layout from an Image. *International Journal on Computer Vision*, 75(1):151–172.

Holynski, A. and Kopf, J. (2018). Fast depth densification for occlusion-aware augmented reality. *ACM Transactions on Graphics (TOG)*.

Horn, B. and Brooks, M. (1989). *Shape from Shading*. MIT Press.

Horn, B. K. (1975). Obtaining shape from shading information. *The psychology of computer vision*, pages 115–155.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*.

Howard, I. P. (2012). *Perceiving in Depth: Volume 1 Basic Mechanisms*. Oxford University Press.

Huang, G., Liu, Z., Pleiss, G., Van Der Maaten, L., and Weinberger, K. (2019). Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017a). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Huang, H., He, R., Sun, Z., and Tan, T. (2017b). Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *ICCV*.

Huang, P., Matzen, K., Kopf, J., Ahuja, N., and Huang, J. (2018a). Deepmvs: Learning Multi-View Stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830.

Huang, S., Qi, S., Zhu, Y., Xiao, Y., Xu, Y., and Zhu, S.-C. (2018b). Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image. In *European Conference on Computer Vision*.

Huynh, L., Nguyen, P., Matas, J., Rahtu, E., and Heikkilä, J. (2022). Lightweight monocular depth with a novel neural architecture search method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3643–3653.

Intille, S. and Bobick, A. (1994). Disparity-Space Images and Large Occlusion Stereo. In *European Conference on Computer Vision*.

Izadinia, H., Shan, Q., and Seitz, S. M. (2017). Im2CAD. In *IEEE Conference on Computer Vision and Pattern Recognition*.

J. Dai, H. Q., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable Convolutional Networks. In *International Conference on Computer Vision*.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*.

Jeon, J. and Lee, S. (2018). Reconstruction-Based Pairwise Depth Dataset for Depth Image Enhancement Using CNN. In *European Conference on Computer Vision*.

Jiang, H. and Xiao, J. (2013). A Linear Approach to Matching Cuboids in RGBD Images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jiang, X., Pendu, M. L., and Guillemot, C. (2018). Depth Estimation with Occlusion Handling from a Sparse Set of Light Field Views. *IEEE International Conference on Image Processing*, pages 634–638.

Jiao, J., Cao, Y., Song, Y., and Lau, R. W. H. (2018). Look Deeper into Depth: Monocular Depth Estimation with Semantic Booster and Attention-Driven Loss. In *European Conference on Computer Vision*.

Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174.

Kanade, T. and Okutomi, M. (1994). A Stereo Matching Algorithm with an Adaptative Window: Theory and Experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932.

Kang, E., Chang, W., Yoo, J., and Ye, J. C. (2018). Deep convolutional framelet denosing for low-dose CT via wavelet residual network. *IEEE Transactions on Medical Imaging*, 37(6):1358–1369.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kirillov, A., Wu, Y., He, K., and Girshick, R. (2020). Pointrend: Image segmentation as rendering. In *CVPR*.

Klingner, M., Termöhlen, J.-A., Mikolajczyk, J., and Fingscheidt, T. (2020). Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *European Conference on Computer Vision*, pages 582–600. Springer.

Kluger, F., Ackermann, H., Brachmann, E., Yang, M. Y., and Rosenhahn, B. (2021). Cuboids Revisited: Learning Robust 3D Shape Fitting to Single RGB Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13070–13079.

Koch, T., Liebel, L., Fraundorfer, F., and Körner, M. (2018). Evaluation of CNN-Based Single-Image Depth Estimation Methods. In *European Conference on Computer Vision*.

Koch, T., Liebel, L., Körner, M., and Fraundorfer, F. (2020). Comparison of monocular depth estimation methods using geometrically relevant metrics on the ibims-1 dataset. *CVIU*.

Kopf, J., Matzen, K., Alsisan, S., Quigley, O., Ge, F., Chong, Y., Patterson, J., Frahm, J.-M., Wu, S., Yu, M., Zhang, P., He, Z., Vajda, P., Saraf, A., and Cohen, M. F. (2020). One shot 3d photography. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4).

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114.

Ladicky, L., Shi, J., and Pollefeys, M. (2014). Pulling Things Out of Perspective. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper Depth Prediction with Fully Convolutional Residual Networks. In *International Conference on 3D Vision*, pages 239–248.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional Networks for Images, Speech, and Time Series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Lee, D. C., Gupta, A., Hebert, M., and Kanade, T. (2010). Estimating Spatial Layout of Rooms Using Volumetric Reasoning About Objects and Surfaces. In *Advances in Neural Information Processing Systems*.

Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization Using Optimization. In *ACM Transactions on Graphics*.

Li, J., Klein, R., and Yao, A. (2017). A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images. In *International Conference on Computer Vision*, pages 3392–3400.

Li, Q., Shen, L., Guo, S., and Lai, Z. (2020a). Wavelet integrated cnns for noise-robust image classification. In *CVPR*.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). Pointcnn: Convolution on x-transformed points. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Li, Y., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2016). Deep Joint Image Filtering. In *European Conference on Computer Vision*.

Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. (2011). GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12.

Li, Z., Shafiei, M., Ramamoorthi, R., Sunkavalli, K., and Chandraker, M. (2020b). Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2475–2484.

Li, Z. and Snavely, N. (2018). Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050.

Liu, C., Kim, K., Gu, J., Furukawa, Y., and Kautz, J. (2019). Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459.

Liu, C., Yang, J., Ceylan, D., Yumer, E., and Furukawa, Y. (2018). PlaneNet: Piece-Wise Planar Reconstruction from a Single RGB Image. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Liu, F., Shen, C., and Lin, G. (2015). Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170.

Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., and Theobalt, C. (2020a). Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663.

Liu, L., Liu, J., Yuan, S., Slabaugh, G., Leonardis, A., Zhou, W., and Tian, Q. (2020b). Wavelet-based dual-branch network for image demoireing. In *ECCV*.

Liu, Y., Shun, C., Wang, J., and Shen, C. (2020c). Structured knowledge distillation for dense prediction. *PAMI*.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *ICCV*.

Lobay, A. and Forsyth, D. (2006). Shape from Texture Without Boundaries. *International Journal on Computer Vision*, 67(1):71–91.

Lopes, A., Souza, R., and Pedrini, H. (2022). A survey on rgb-d datasets. *arXiv preprint arXiv:2201.05761*.

Lowe, D. (1999). Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, pages 1150–1157.

Luo, C., Li, Y., Lin, K., Chen, G., Lee, S.-J., Choi, J., Yoo, Y. F., and Polley, M. O. (2020a). Wavelet synthesis net for disparity estimation to synthesize dslr calibre bokeh effect on smartphones. In *CVPR*.

Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., and Yuille, A. (2019). Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2624–2641.

Luo, X., Zhang, J., Hong, M., Qu, Y., Xie, Y., and Li, C. (2020b). Deep wavelet network with domain adaptation for single image demoireing. In *CVPR Workshops*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning*.

Mahjourian, R., Wicke, M., and Angelova, A. (2018). Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*.

Martin, D., Fowlkes, C., and Malik, J. (2004). Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5).

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A Database of Human Segmented Natural Images and Its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *International Conference on Computer Vision*.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE.

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048.

McCormac, J., Handa, A., Leutenegger, S., and J.Davison, A. (2017). Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*.

Ming, Y., Meng, X., Fan, C., and Yu, H. (2021). Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33.

Monszpart, A., Mellado, N., Brostow, G. J., and Mitra, N. J. (2015). RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103:1–103:12.

Murez, Z., van As, T., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. (2020). Atlas: End-to-end 3d scene reconstruction from posed images. In *ECCV*.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$.

Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). KinectFusion: Real-Time Dense Surface Mapping and Tracking.

Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., and Yang, Y.-L. (2019). Hologan: Unsupervised learning of 3d representations from natural images. In *The IEEE International Conference on Computer Vision (ICCV)*.

Nguyen-Phuoc, T. H., Richardt, C., Mai, L., Yang, Y., and Mitra, N. (2020). Blockgan: Learning 3d object-aware scene representations from unlabelled images. *Advances in Neural Information Processing Systems*, 33:6767–6778.

Niu, C., Li, J., and Xu, K. (2018). Im2struct: Recovering 3d shape structure from a single rgb image. In *Computer Vision and Pattern Regognition (CVPR)*.

Owen, B. (2007). A Robust Hybrid of Lasso and Ridge Regression. *Contemp. Math.*, 443.

Pan, J., Han, X., Chen, W., Tang, J., and Jia, K. (2019). Deep mesh reconstruction from single rgb images via topology modification networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9964–9973.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Paschalidou, D., Ulusoy, A. O., and Geiger, A. (2019). Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Paschalidou, D., van Gool, L., and Geiger, A. (2020). Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., and Geiger, A. (2020). Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer.

Poggi, M., Aleotti, F., Tosi, F., and Mattoccia, S. (2018a). Towards real-time unsupervised monocular depth estimation on cpu. In *IROS*.

Poggi, M., Tosi, F., Batsos, K., Mordohai, P., and Mattoccia, S. (2021). On the synergies between machine learning and binocular stereo for depth estimation from images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Poggi, M., Tosi, F., and Mattoccia, S. (2018b). Learning Monocular Depth Estimation with Unsupervised Trinocular Assumptions. In *International Conference on 3D Vision*, pages 324–333.

Qi, C., Su, H., Mo, K., and Guibas, L. (2017a). Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Qi, C., Yi, L., Su, H., and Guibas, L. (2017b). Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656.

Qi, X., Liao, R., Liu, Z., Urtasun, R., and Jia, J. (2018). Geonet: Geometric Neural Network for Joint Depth and Surface Normal Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. (2019). On the spectral bias of neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR.

Ramamonjisoa, M., Du, Y., and Lepetit, V. (2020). Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *CVPR*.

Ramamonjisoa, M. and Lepetit, V. (2019). Sharpnet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. In *ICCV Workshop*.

Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*.

Ranjan, A., Jampani, V., Kim, K., Sun, D., Wulff, J., and Black, M. J. (2019). Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*.

Ren, X. and Bo, L. (2012). Discriminatively Trained Sparse Code Gradients for Contour Detection. In *Advances in Neural Information Processing Systems*.

Ren, X., Fowlkes, C., and Malik, J. (2006). Figure/ground Assignment in Natural Images. In *European Conference on Computer Vision*.

Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586.

Roberts, L. G. (1963). *Machine Perception of Three-Dimensional Solids*. PhD thesis, Massachusetts Institute of Technology.

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. (2021). Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, abs/1505.04597.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323:533–536.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*.

Sarbolandi, H., Lefloch, D., and Kolb, A. (2015). Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 139:1–20.

Saxena, A., Chung, S. H., and Ng, A. Y. (2006). Learning Depth from Single Monocular Images. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems*, pages 1161–1168. MIT Press.

Saxena, A., Sun, M., and Ng, A. (2009). Make3D: Learning 3d Scene Structure from a Single Still Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840.

Saxena, A., Sun, M., and Ng, A. Y. (2009). Make3D: Learning 3D Scene Structure from a Single Still Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):824–840.

Scharstein, D. and Szeliski, R. (2002). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal on Computer Vision*, 47(1/2/3):7–42.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Comput. Graph. Forum*, 26:214–226.

Schönberger, J. and Frahm, J. (2016). Structure-From-Motion Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Schönberger, J., Zheng, E., Pollefeys, M., and Frahm, J. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision*.

Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A. (2020). Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166.

Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528.

Sengupta, S., Gu, J., Kim, K., Liu, G., Jacobs, D. W., and Kautz, J. (2019). Neural inverse rendering of an indoor scene from a single image. In *International Conference on Computer Vision (ICCV)*.

Shao, T., Monszpart, A., Zheng, Y., Koo, B., Xu, W., Zhou, K., and Mitra, N. (2014). Imagining the Unseen: Stability-based Cuboid Arrangements for Scene Understanding. *ACM SIGGRAPH Asia 2014*. * Joint first authors.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. (2020). Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shi, W. and Rajkumar, R. (2020). Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719.

Shih, M.-L., Su, S.-Y., Kopf, J., and Huang, J.-B. (2020). 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., and Zollhofer, M. (2019). Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446.

Skiena, S. (2010). *The Algorithm Design Manual*. Springer.

Song, S., Lichtenberg, S. P., and Xiao, J. (2015). SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic Scene Completion from a Single Depth Image. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Spek, A., Dharmasiri, T., and Drummond, T. (2018). CReaM: Condensed real-time models for depth prediction using convolutional neural networks. In *IROS*.

Sterzer, P. and Rees, G. (2006). Perceived size matters. *Nature Neuroscience*, 9(3):302–302.

Subbarao, M. and Surya, G. (1994). Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294.

Sung, M., Kim, V. G., Angst, R., and Guibas, L. (2015). Data-driven Structural Priors for Shape Completion. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*.

Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. Springer.

Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547.

Taubman, D. and Marcellin, M. (2013). *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Springer Publishing Company, Incorporated.

Teng, Q., Chen, Y., and Huang, C. (2018). Occlusion-Aware Unsupervised Learning of Monocular Depth, Optical Flow and Camera Pose with Geometric Constraints. *Future Internet*, 10:92.

The CGAL Project (2022). *CGAL User and Reference Manual*. Number 5.4. CGAL Editorial Board.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*.

Tieleman, T., Hinton, G., et al. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Tomasi, C. and Manduchi, R. (1998). Bilateral Filtering for Gray and Color Images. In *International Conference on Computer Vision*.

Tosi, F., Aleotti, F., Poggi, M., and Mattoccia, S. (2019). Learning monocular depth estimation infusing traditional stereo knowledge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tosi, F., Aleotti, F., Zama Ramirez, P., Poggi, M., Salti, S., Di Stefano, L., and Mattoccia, S. (2020). Distilled semantics for comprehensive scene understanding from videos. In *IEEE Conference on Computer Vision and Pattern Recognition*. CVPR.

Tulsiani, S., Su, H., Guibas, L. J., Efros, A. A., and Malik, J. (2017). Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Regognition (CVPR)*.

Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., and Geiger, A. (2017). Sparsity invariant CNNs. In *3DV*.

Unser, M. and Blu, T. (2003). Mathematical properties of the jpeg2000 wavelet filters. *IEEE Transactions on Image Processing*, 12(9):1080–1090.

Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F. Z., Daniele, A. F., Mostajabi, M., Basart, S., Walter, M. R., and Shakhnarovich, G. (2019). DIODE: A Dense Indoor and Outdoor DEpth Dataset.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., and Fragkiadaki, K. (2017). Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.

Wang, G., Liang, X., and Li, F. W. B. (2018a). DOOBNet: Deep Object Occlusion Boundary Detection from an Image. *CoRR*, abs/1806.03772.

Wang, L., Zhang, J., Wang, O., Lin, Z., and Lu, H. (2020). Sdc-depth: Semantic divide-and-conquer network for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 541–550.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. (2018b). Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67.

Wang, P., Shen, X., Lin, Z. L., Cohen, S., Price, B. L., and Yuille, A. L. (2015). Towards Unified Depth and Semantic Prediction from a Single Image. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809.

Wang, P., Shen, X., Russell, B., Cohen, S., Price, B., and Yuille, A. L. (2016). SURGE: Surface Regularized Geometry Estimation from a Single Image. In *Advances in Neural Information Processing Systems*, pages 172–180.

Wang, P. and Yuille, A. (2016). DOC: Deep Occlusion Estimation from a Single Image. In *European Conference on Computer Vision*.

Wang, T., Efros, A. A., and Ramamoorthi, R. (2016). Depth Estimation with Occlusion Modeling Using Light-Field Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2170–2181.

Wang, Z., Philion, J., Fidler, S., and Kautz, J. (2021). Learning indoor inverse rendering with 3d spatially-varying lighting. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Watson, J., Firman, M., Brostow, G. J., and Turmukhambetov, D. (2019). Self-supervised monocular depth hints. In *ICCV*.

Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., and Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*.

Wen, W., Wu, C., Wang, Y., Chen, Y., and Li, H. (2016). Learning structured sparsity in deep neural networks. In *NeurIPS*.

Witkin, A. P. (1981). Recovering surface shape and orientation from texture. *Artificial intelligence*, 17(1-3):17–45.

Wofk, D., Ma, F., Yang, T.-J., Karaman, S., and Sze, V. (2019). FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *ICRA*.

Woodham, R. (1980). Photometric Method for Determining Surface Orientation from Multiple Images. *Optical Engineering*, 19(1):139–144.

Wu, H., Zheng, S., Zhang, J., and Huang, K. (2018a). Fast End-To-End Trainable Guided Filter. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., and Tenenbaum, J. (2017). Marrnet: 3d shape reconstruction via 2.5 d sketches. *Advances in neural information processing systems*, 30.

Wu, J., Zhang, C., Zhang, X., Zhang, Z., Freeman, W. T., and Tenenbaum, J. B. (2018b). Learning shape priors for single-view 3d completion and reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–662.

Wu, Z. and Li, L. (1988). A Line-Integration Based Method for Depth Recovery from Surface Normals. *Computer Vision, Graphics, and Image Processing*, 43(1):53–66.

Xian, K., Zhang, J., Wang, O., Mai, L., Lin, Z., and Cao, Z. (2020). Structure-guided ranking loss for single image depth prediction. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xie, J., Girshick, R., and Farhadi, A. (2016). Deep3D: Fully Automatic 2D-To-3d Video Conversion with Deep Convolutional Neural Networks. In *European Conference on Computer Vision*, pages 842–857.

Xiong, J., Hsiang, E.-L., He, Z., Zhan, T., and Wu, S.-T. (2021). Augmented reality and virtual reality displays: emerging technologies and future perspectives. *Light: Science & Applications*, 10(1):1–30.

Xiong, Y. and Shafer, S. A. (1993). Depth from focusing and defocusing. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 68–73. IEEE.

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Xu, D., Ricci, E., Ouyang, W., Wang, X., and Sebe, N. (2017). Multi-Scale Continuous CRFs as Sequential Deep Networks for Monocular Depth Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Xu, D., Ricci, E., Ouyang, W., Wang, X., and Sebe, N. (2018a). Monocular Depth Estimation Using Multi-Scale Continuous CRFs as Sequential Deep Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. (2018b). Spidercnn: Deep learning on point sets with parameterized convolutional filters. *arXiv preprint arXiv:1803.11527*.

Yang, M., Wu, F., and Li, W. (2020). Waveletstereo: Learning wavelet coefficients of disparity map in stereo matching. In *CVPR*.

Yang, T.-J., Howard, A., Chen, B., Zhang, X., Go, A., Sandler, M., Sze, V., and Adam, H. (2018a). Netadapt: Platform-aware neural network adaptation for mobile applications. In *ECCV*.

Yang, Z., Wang, P., Wang, Y., Xu, W., and Nevatia, R. (2018b). Lego: Learning edge with geometry all at once by watching videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 225–234.

Yang, Z., Xu, W., Zhao, L., and Nevatia, R. (2018c). Unsupervised Learning of Geometry From Videos With Edge-Aware Depth-Normal Consistency. In *AAAI*.

Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. (2018). Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783.

Ye, J., Lu, X., Lin, Z., and Wang, J. Z. (2018). Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv:1802.00124*.

Yi, K. M., Trulls, E., Lepetit, V., and Fua, P. (2016). LIFT: Learned Invariant Feature Transform. In *European Conference on Computer Vision*.

Yin, W., Liu, Y., Shen, C., and Yan, Y. (2019). Enforcing Geometric Constraints of Virtual Normal for Depth Prediction. In *International Conference on Computer Vision*.

Yin, W., Zhang, J., Wang, O., Niklaus, S., Mai, L., Chen, S., and Shen, C. (2021). Learning to recover 3d scene shape from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 204–213.

Yin, Z. and Shi, J. (2018). GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. (2019). Slimmable neural networks. In *ICLR*.

Yu, Y. and Smith, W. A. (2019). Inverserendernet: Learning single image inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zama Ramirez, P., Poggi, M., Tosi, F., Mattoccia, S., and Di Stefano, L. (2018). Geometry meets semantics for semi-supervised monocular depth estimation. In *Asian Conference on Computer Vision*, pages 298–313. Springer.

Zhang, R., Tsai, P.-S., Cryer, J. E., and Shah, M. (1999). Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706.

Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*.

Zhang, Y. and Funkhouser, T. (2018). Deep depth completion of a single rgb-d image. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Y., Song, S., Yumer, E., Savva, M., Lee, J.-Y., Jin, H., and Funkhouser, T. (2017). Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., and Yang, J. (2019). Pattern-Affinitive Propagation Across Depth, Surface Normal and Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhao, C., Sun, Q., Zhang, C., Tang, Y., and Qian, F. (2020). Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid Scene Parsing Network. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhao, Y. and Zhu, S.-C. (2013). Scene Parsing by Integrating Function, Geometry and Appearance Models. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. (2017). Unsupervised Learning of Depth and Ego-Motion from Video. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499.

Zhu, R., Li, Z., Matai, J., Porikli, F., and Chandraker, M. (2022). Irisformer: Dense vision transformers for single-image inverse rendering in indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2822–2831.

Zhu, S., Brazil, G., and Liu, X. (2020). The edge of depth: Explicit constraints between segmentation and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13116–13125.

Zou, C., Guo, R., Li, Z., and Hoiem, D. (2019). Complete 3D Scene Parsing from an RGBD Image. *International Journal on Computer Vision*.

Zou, C., Yumer, E., Yang, J., Ceylan, D., and Hoiem, D. (2017). 3d-prnn: Generating shape primitives with recurrent neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*.

Zwald, L. and Lambert-Lacroix, S. (2012). The Berhu Penalty and the Grouped Effect.