The Usage of Quadtree in Deep Neural Networks to Represent Data For Navigation From a Monocular Camera

Daniel Braun

UNIVERSITÉ DE FRANCHE-COMTÉ

THESE DE DOCTORAT DE L'ETABLISSEMENT UNIVERSITE BOURGOGNE FRANCHE-COMTE

PREPAREE A L'UNIVERSITE DE BOURGOGNE

Ecole doctorale nº ED37
Sciences Physiques pour l'Ingénieur et Microtechniques (SPIM)

Doctorat en Instrumentation et Informatique de l'Image

Par

M. Daniel BRAUN

L'utilisation de Quadtree dans les Réseaux de Neurones Profonds pour Représenter les Données pour la Navigation à partir d'une Caméra Monoculaire.

The Usage of Quadtree in Deep Neural Networks to Represent Data for Navigation from a Monocular Camera.

Thèse présentée et soutenue à Le Creusot, le 14 décembre 2022

Composition du Jury :

M. Thierry CHATEAU	Professeur des universités, Université Clermont Auvergne	Président
Mme Sylvie CHAMBON	Maître de conférences, INP Toulouse - ENSEEIHT	Rapporteur
M. Vincent FREMONT	Professeur des universités, Ecole Centrale de Nantes	Rapporteur
M. Cédric DEMONCEAUX	Professeur des universités, Université de Bourgogne-Franche-Comté (Dijon) - Laboratoire ImViA	Directeur de thèse
M. Pascal VASSEUR	Professeur des universités, Université de Picardie Jules Verne	Codirecteur de thèse
M. Olivier MOREL	Maître de conférences, Université de Bourgogne-Franche -Comté (Dijon) - Laboratoire ImViA	Codirecteur de thèse

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Daniel Braun

04-02-2023

Dedication

I would like to dedicate this work to my three-year-old son, Léonce, and to my little angle, Ézechiel, who left us far too soon.

Acknowledgments

I would like to express my sincerest appreciation to my advisors, Cédric Demonceaux, Olivier Morel and Pascal Vasseur, for their trust, guidance, support, and encouragement throughout this project. Their expertise and feedback have played a crucial role in shaping the direction and outcome of my project.

I would also like to thank the members of my thesis jury, Thierry Chateau, Sylvie Chambon, and Vincent Frémont, for their time, effort, and dedication in reviewing my thesis. Their acceptance to be part of the jury has been a great honor.

I would like to extend my appreciation to my colleagues at ImViA ViBot for their ongoing support throughout my research, both in professional matters and during coffee breaks, whether it was in person or remotely.

I would like to thank my friends and family for their support and the interest they showed of my work, even though some of them had a hard time understanding what I was doing.

I would like to extend my warmest gratitude to my parents for their unwavering love and support throughout my life. Their hard work and dedication in raising me has made it possible for me to be where I am today. I am forever grateful for everything they have done for me and without them, this achievement would not have been possible.

Finally, I would like to offer my heartfelt thanks to my wife who willingly accepted to join me in this journey. She has been the most supportive and understanding during the ups and downs of my mood, making this experience so much more enjoyable. She has been my rock and gave me everything I could have ever wanted, including making me a husband and a father. Thank you for everything.

Abstract

Depth acquisition represents a key element for navigation tasks. It is, therefore, one of the major research topics in computer vision. Many approaches have been developed to address this problem by constructing the depth from series of images. However, there is a minimal case proposing a prediction from a single image, made possible with the emergence of deep learning approaches. The latter makes it possible to consider a reduction in both hardware and computing time costs, which is beneficial for embedded systems. However, network architecture remains a heavy process requiring a lot of GPU memory. Few approaches have proposed addressing this problem by developing lightweight architectures, allowing real-time execution. We propose to investigate this problem from another angle, consisting in carefully selecting the operations to execute rather than lightening the architecture. It is based on the Quadtree Generating Networks framework which takes advantage of sparse convolutions to only operate necessary operations to generate the quadtree, thus reducing the computational cost. This method, initially developed for semantic segmentation, will be applied in this study to data acquisition problems for navigation. Namely, the image segmentation for obstacle avoidance and the generation of compressed depth maps into quadtree. It will be demonstrated, through a series of experiments, that the quadtree compression allows a significant reduction of the memory requirement with a limited loss of accuracy. The level of compression of the prediction is fully adjustable for the depth estimation, making it adaptable to all situations.

The thesis is written in english and is 92 pages long, including 6 chapters, 35 figures and 13 tables.

List of Abbreviations and Terms

CNN	Convolutional Neural Network.
FCN	Fully Convolutional Network.
FLOPs	Floating-points operations.
GPU	Graphics Processing Units.
GT	Ground truth.
IMU	Inertial Measurement Unit.
LiDAR	The Light Detection and Ranging is a laser scanning sensor to acquire a 3D point cloud of the surrounding environment.
OGN	Octree Generating Networks.
QGN	Quadtree Generating Networks.
RGB	Type of camera acquiring colors encoded as three channels: red, green and blue.
RGBD	Type of camera providing both color (RGB) and depth (D) data.
RMSE	Root Mean Square Error.
SfM	Structure from Motion.
SLAM	Simultaneous Localization And Mapping.
SSIM	Structural Similarity Index Measure.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivations

Autonomous navigation is one of the main research topic in robotics and computer vision. It consists of acquiring data of the surrounding environment and of using it to decide on the most appropriate path to follow. The strategy may vary depending on the objective, but whether it is to find the shortest, fastest or safest route, the decision has to be made based on its perception of the world. Therefore, it is preferable to have the most appropriate solution, depending on the desired application. The choice of solution will also vary based on the available sensors on the systems, which is largely defined by the dimensions and the budget of the mobile system. The same set of sensors cannot be implemented whether it is on a car or on a drone which has strict size and weight limitations.

From this point, the choice of approach will diverge between solution proposing to fuse information from a set of sensors and those extracting the most information available from a single sensor. The objective of the latter being to be lighter, to offer an interesting trade-off between computation cost and accuracy, while also being cheaper. Single sensor systems are either using LiDAR[1] or RGB[2] camera. LiDAR is a 360 degree laser scanner, mainly based on the time of flight (ToF) technology, generating a three-dimensional point cloud of the scene, providing an accurate sampling of obstacles location. It is on of the most reliable solution to acquire depth information and is doing it in every directions making it even more interesting. Yet, it remains an expansive sensor with some limitations, such as the point cloud being too sparse for some applications and the negative effect of

[1] Acronym for "light detection and ranging".
[2] Acronym for "Red, Green and Blue" corresponding to the three channels composing the image color.

the sunlight on the acquisition [41]. Therefore, the use of camera is generally preferred in single sensor application, as it is a cheaper and represents a more versatile solution. However, as it only produces a two dimensional information, the extraction of depth, when using geometrical algorithm approaches, requires the acquisition of two or more images of the scene as theorized by Hartley and Zisserman in [38]. It is performed in practice by either adding a second camera in order to form a calibrated stereo system, or by estimating the relative motion of the camera between two time stamps. Both solutions can be used to navigate by successively aggregating scene information using SLAM [74] or SfM [73] algorithms.

For the last decade, deep learning approaches have progressively conquered the computer vision research fields [1]. Thus, new methods, in today's literature, not featuring deep learning became the exception. Indeed, the neuron architectures have demonstrated there capability to efficiently solve and outperform numerous problems that were once accomplished with *classical* algorithms. Most importantly, they expanded the research to new fields that were hardly achievable before, among which are semantic segmentation [29] and monocular depth estimation [22]. Both topics are transforming information from an single image into something else they have learnt to predict through a training procedure. The semantic segmentation consists of classifying information in the image based on its type, which is highly valuable to increase the scene understanding. The monocular depth estimation consists, as indicated by its name, of predicting depth from a single image. It is normally an ill-posed problem, as two images are usually required to triangulate points as explained before. But through learning, and in similar image configurations, the network manages to estimate distance of objects by considering the global context of the scene. Both approaches are interpreting and solving there tasks by mimicking what they learned during training. The downside being the method performances depends on the training set and may produce aberrant results when processing images from outside the scope. Depth prediction methods trained on outdoor urban scenes may misinterpret distances on indoor images due to the layout differences.

Even if those monocular depth prediction methods are producing satisfying results, they are scarcely used for real-time navigation [22]. This limitation can be explained by different factors. As discussed earlier, the quality of the prediction depend on the datasets used for training, meaning the quality might diminish to become unreliable in a different environments. Deep learning networks can also be expansive to compute, depending on their network size. Some of those approaches are focused on dense 3D reconstruction and are essentially working on improving the network architecture or the training procedure to outperform the current state of the art [10, 60, 54]. Yet, others addressed this issue by developing lightweight solutions specifically designed for real-time applications [80, 16].

But it necessarily comes at the expense of prediction accuracy or input image resolution. Subsequently, it all comes to a trade-off between the computation cost and the accuracy of the prediction.

1.2 Scope and Objectives

This thesis is part of the ANR CLARA, which gathers different research projects whose aim at solving problems related to the navigation of autonomous UAVs. The tasks have been divided between perception and control to achieve such goal. The objective of this thesis is to explore solutions for perception from RGB camera. The choice has been made to research solutions for compressed representation. It emerged from the observation that today's research on depth acquisition is mostly oriented toward higher accuracy rather than time efficiency for real time application. In additions, works related to sparsity in the network architecture is expanding and represents a promising alternative solution to dense neural networks. They assume that by carefully selecting the operations to perform, methods are able to achieve similar results at cheaper price [47].

It is proposed in this work to explore an alternative solution consisting of introducing sparsity in the network to generate a compressed prediction organized into quadtree. It permits to reduce the computation cost, thus to accelerate the execution time, while keeping high resolution at the necessary locations in the image. Indeed, a quadtree is a hierarchical tree data structure composed of nodes. Each node stores the data and position on the tree, which indicates its localization on the image and the size of the area it represents. For the specific case of images, they represent square area in the image and can then be subdivided into four smaller square area of equal size. The advantage of the tree structure is that it allows information representing of different size in the image to coexist, permitting to compress the data while preserving important information at higher resolution. Quadtrees are normally used to compressed dense information, inducing the higher resolution has to be known. With the usage of the Quadtree Generating Networks (QGN) framework [14], it is possible to directly infer a quadtree by skipping the computation of all data that will not be present in the compressed representation.

1.3 Contributions

The work presented in this manuscript has led to the publication of the following three papers:

- "N-QGN: Navigation Map from a Monocular Camera using Quadtree Generating

Networks."
Daniel Braun, Olivier Morel, Pascal Vasseur & Cédric Demonceaux.
In 2022 IEEE International Conference on Robotics and Automation (ICRA).

- "Quadtree Segmentation Network for Obstacle Avoidance in Monocular Navigation."
  Daniel Braun, Olivier Morel, Pascal Vasseur & Cédric Demonceaux.
  In 2022 IEEE International Conference on Intelligent Transportation Systems (ITSC).
- "Optimized Quadtree Depth Estimation from a Monocular Camera."
  Daniel Braun, Olivier Morel, Pascal Vasseur & Cédric Demonceaux.
  Under Review at IEEE Robotics and Automation Letters.

1.4 Document Organization

The document is organized as follows:

Chapter 2 provides a condensed analysis of the literature and how it relates to the presented works. Topics related to segmentation, monocular depth prediction and the use of quadtree representation in deep learning will be covered.

Chapter 3 presents the quadtree segmentation work applied to obstacle avoidance. It allows a first overview of the use of Quadtree Generating Networks as well as a reflection on unsupervised segmentation.

Chapter 4 focuses on the interest of Quadtree Generating Networks applied to the compression of continuous information such as depth data. It explores solutions to efficiently compress information while limiting the loss of precision while being trainable in an self-supervised manner.

Chapter 5 proposes to combine the best of both worlds from the two previous approaches while improving the results. It presents how the learning guidance and the prediction of the optimum quadtree structure by the network allow to improve the results in terms of accuracy and stability. Improvements on the implementation have also been made to achieve a real-time inference.

Chapter 6 is the conclusion of these research works and proposes perspectives on the evolution as well as future works to be considered.

The choice was made to separate the chapters by application as it permits to follow the progression path and the improvements made in the use of Quadtree Generating Network, which acts as a thread through the document.

Chapter 2

Related Literature

This chapter consists of an overview of the works from the literature related to our studies. It begins with a presentation of the quadtree data structures applied to computer vision and its benefits for deep learning approaches. It continues with the evolution on the monocular depth estimation and its interests for navigation. The last section presents the evolution of image segmentation approaches using deep learning, the main benchmark datasets and the perspectives toward self-supervised training and real-time inference.

2.1 Quadtree Data Structure for Deep Learning

This section presents the dual benefit of using quadtree data structure in deep learning approaches. The first part is redefining the quadtree and octree representation in computer vision. The second part presents their applications to navigation. The third part is presenting their benefits for deep learning.

2.1.1 Quadtree and Octree Representation

The quadtree is a hierarchical tree data structure [66] that is deeply tight to the image processing field since it was first introduced nearly 40 years ago by Samet et al. [66]. It consists of representing information in an image as graph of interconnected nodes, as illustrated in Figure 1. It results in a tree following a hierarchical structure, as the nodes location in the tree defines their position in the image and the area they represent. As opposed to the representation in the form of a grid of pixels, the quadtree data structure permits to information of different sizes to coexist. This allows a lot of freedom to compress information in areas with little texture or low interest.

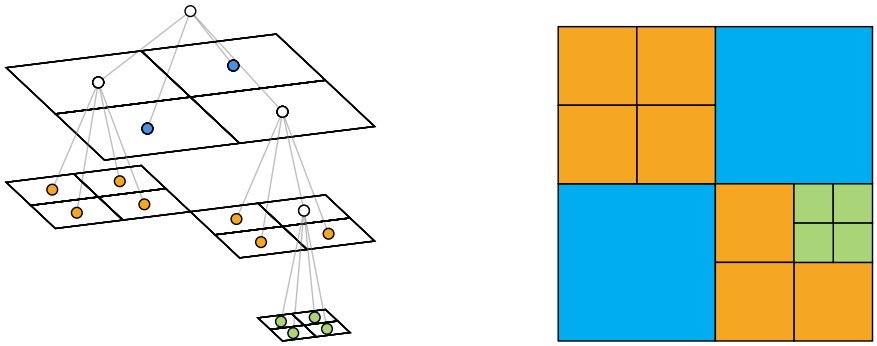

Figure 1. Representation of the quadtree data structure applied to images. On the left is a perspective view representing the subdivision of the quadtree nodes. On the right is the recomposed image.

The quadtree, as defined by [14], can be described by the following equation 2.1. It is a set of N nodes, of which the i^{th} node is characterized by its depth level l_i in the tree, its centroid coordinates (x_i, y_i) and its value v_i.

$$Q = \{(l_i, x_i, y_i, v_i) \mid i = \{1, .., N\}\} \tag{2.1}$$

The depth level of the tree corresponds in the image as the size of the square area represented by its node. Therefore, the quadtree will often be separated by its nodes depth level and noted $Q_l = \{Q \mid l_i = l\}$. All these nodes form an image of the scene at a given resolution. In general, the quadtree can be seen as a multi-resolution sparse representation of the image, as in Figure 1.

Another popular tree data structure is the octree [53], which corresponds to the 3-dimensions counterpart of the quadtree. It follows the exact same logic, except it represents 3D volumes and the nodes are subdivided into 8 sub-cubes instead of 4 sub-squares. They are, as such, the compressed representation of the voxel[1] grid. A similar definition can be applied using 3D coordinates for its centroid (x_i, y_i, z_i) to form the octree O as follows:

$$O = \{(l_i, x_i, y_i, z_i, v_i) \mid i = \{1, .., N\}\} \tag{2.2}$$

[1] unit cube on a 3-dimensional regular grid. It is the equivalent in 3D of a pixel.

a) Intensity Image b) Depth Map

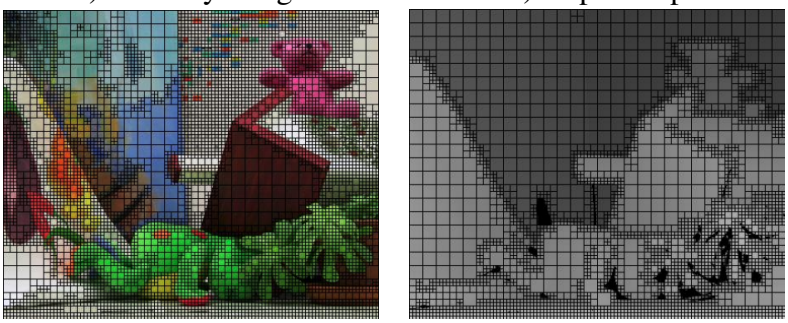

Figure 2. An intensity image and its corresponding depth map, from Middlebury Stereo Datasets [68], converted into quadtree. As suggested by [79], pixels in the depth map are generally more compressed than in the intensity image. It implies all pixels inside a same node necessarily share the same depth information.

2.1.2 Quadtree for Navigation

The data structure similarity between octree and quadtree implies the progression and use of these two approaches are linked. Even though, octree approaches have been more used for navigation because of the huge benefit brought by compression. Methods such as Octomap [40] have allowed to generate large occupancy maps. Point clouds are aggregated on navigation sequences to update the probable locations of obstacles on the map. Nodes are subdivided if higher resolution is necessary to describe the volume, but could also be merged if all the sub-nodes share the same values. Therefore, the addition of new information does not necessary induce a growth of stored data in the map.

Applications using the quadtree data structure for navigation remain limited. It is often preferred to use the image according to a regular pixel grid, even if it means working with smaller images. Indeed, the conversion to quadtree is necessarily done in post-processing, adding computation time. This implies that the gain in speed induced by the compression must be significant to justify its use. Besides, the manipulation of data in tree form is not trivial and requires the use of algorithms optimized for these operations [70]. However, one notable method [79] proposed to use quadtree compression for real-time dense mapping. They started from the observation that in most cases, the same area of the image is more compressed when it represents a depth value than a light intensity, as illustrated in Figure 2. Therefore, the quadtree compression of the input image does not degrade the depth prediction. This allows significant compression of the input images - which enables real-time depth prediction using the dynamic belief propagation [87]. This method consists of iteratively estimating and updating the depth values by combining information from nearby previous frames. Working with quadtree structure permits to only estimate a sparse depth on the node centroids. The full depth map can then be reconstructed

using interpolations.

2.1.3 Quadtree in Deep Learning

2.1.3.1 Sparse Convolutions

Convolutional neural networks (CNN) consist in processing the input data of the network through a sequence of convolution operations. The data are organized in the form of tensors, corresponding to dense multi-dimensional arrays. They are generally organized as $B \times C \times H \times W$ when the input is two-dimensional and $B \times C \times H \times W \times D$ when it is three-dimensional. B corresponds to the batch size, i.e. the number of independent data processed in parallel. C is the number of channels and H, W, D are respectively the height, the width and the depth of the data.

Convolution-based approaches in computer vision represent, along with transformers [45], the vast majority of the state of the art in deep learning. These methods have in common that they manipulate dense data structures, which is not adequate to the use of the quadtree. However, the need arose to apply convolutions on sparse data. Whether the data is natively sparse [75] or rendered sparse through the network [58]. This is made possible with the development of sparse convolutions [35] which allow to ignore locations containing no data. In practice, this corresponds to pixels containing zero values, which are thus removed from the activation map. It is the latter that defines the data to be taken into account in the calculation of convolutions. Subsequently, less information are to be computed, reducing the networks complexity and the floating-points operations (FLOPs).

2.1.3.2 Quadtree Generating Networks

The Quadtree Generating Networks (QGN) [79] and its big brother Octree Generating Networks (OGN) [77] are convolutionnal decoder architectures transforming an input dense information into an optimized tree-based representation. These methods are proposing a coarse to fine prediction, where at each step only the set of data that needs to be refined are processed. This procedure implies the coexistence on the same plane information with different dimensions, thus organized as a quadtree/octree structure.

QGN is proposing a quadtree semantic segmentation and OGN a 3d shape prediction with probability outputs being either labeled as *empty*, *filled* or *mixed*. The two methods have in common the *mixed* status, indicating the corresponding node cannot be classified at this resolution and thus have to be subdivided. It is in that sense that tree structure is constructed inside the decoder. Subsequent layers will only processed nodes which received the *mixed* status on the previous step. Therefore, a large amount of computation

Method	**Model**	**Memory** (GB)	**FLOPs** ($\times 10^{12}$)	**mIoU** (%)
Dilatation	DRN-C-42 [89]	3.77	1.07	70.9
	DRN-D-105 [89]	15.15	1.91	75.6
	DeepLabv3 [9]	14.27	1.97	79.3
	CCNet [42]	14.33	1.55	79.8
QGN	dense	5.85	0.48	78.2
	quadtree	3.66	0.25	73.0

Table 1. Results from QGN [79] on the memory consumption, networks complexity and accuracy with respect to state of the art methods. Experiments have been conducted on Cityscapes [20] for input resolution of 2048×1024.

can be saved, significantly reducing the need in memory.

Methods are trained with a ground-truth supervision by optimizing the cross-entropy loss function. The *mixed* status is considered by the network as an extra class, and will implicitly learn to predict the structure through training. Nonetheless, it is acknowledge that the usage as guidance of the ground-truth tree structure during training time greatly improves the performances. The way the pixels are propagated through the decoder is highly dependent on the quality of the prediction. It appeared in practice to be unreliable for training and therefore a ground truth (GT) guidance is preferred. If such data is not available, it is suggested to either pretrain the model to have reliable prediction from the start or to propagate all pixels to achieve a dense equivalent training. For the latter, it will induce highest memory consumption as the prediction will be dense. At inference time, the absence of guidance is no longer an issue, as the network learnt to predict the structure.

As mentioned early, the interest of the method is its capability to reduce the memory consumption compared to dense equivalent methods as demonstrated in QGN. The table 1 is demonstrating this benefit while still achieving top accuracy level. It allows us to situate the method in the context of the state of the art. The method is lighter than the others even in its dense use, i.e. by propagating all the pixels thus without any compression. On the mIoU results, its performances are better than DRN-D-105 [89], while being lighter on the FLOPs and GPU memory use. Beyond that, the method proposes to compress the prediction in quadtree and thus to reduce it even further. Yet, the loss in accuracy is significant, with a loss of 5 points compared to the dense prediction. However, this loss is accompanied by a huge gain on the computational cost and offers a trade-off between lightness and accuracy. It can be compared with DRN-C-42 [89] by proposing equivalent memory use with a reduction of FLOPs and an increase of accuracy.

2.2 Monocular Depth Estimation

2.2.1 Problem Statement

Monocular depth methods consist of proposing a pixel to depth prediction from a single image. This solution emerged with the pioneer work of Eigen et al. [25] who searched for solutions to increase the understanding of massively available images on the web. At the time, numerous solutions already existed to acquire depth from stereo images but the works on single frame were extremely limited. This comes from the fact that the monocular depth prediction is an ill-posed problem with uncertainty arising from the incapability to properly estimate the scale. Three-dimensional data are generally acquired by triangulating points at different locations from a set of images. The use of CNN permits to lift these ambiguities by extracting key features from the image and enables to understand the context and the nature of the information.

Ever since the early work from Eigen et al. [25] the research community of monocular depth estimation kept growing and remains largely active as of today. Each method comes with innovative changes to increase the prediction. These approaches can be categorized into three main families which are the supervised, self-supervised and semi-supervised training. The goal remains unchanged, but the way to achieve it differs. Supervised learning consists of training the network to mimic the GT depth [49]. The optimization loss function is generally based on pixel-wise minimization of the L1, L2, log depth or Berhu functions [56]. But the access to the full GT depth is not always possible and can be expansive to generate. Subsequently, semi-supervised training proposes to integrate external information to enrich learning. This can take several forms, such as the use of synthetic data [90], a LiDAR sensor giving partial depth data [63] or the access to the normal surface [62]. These methods are also geometric based and aims at enforcing the consistency of the external data source. Lastly, some approaches have turned to unsupervised learning which consists in evaluating the capacity of the prediction to verify geometric properties. Indeed, the calculation of a depth map from a set of images is a known geometric property. It is therefore possible to reverse the problem and use the predicted depth to recreate the missing image pair from the first image. The objective is to virtually recreate an existing close image and to evaluate the resulting photometric error [30, 92].

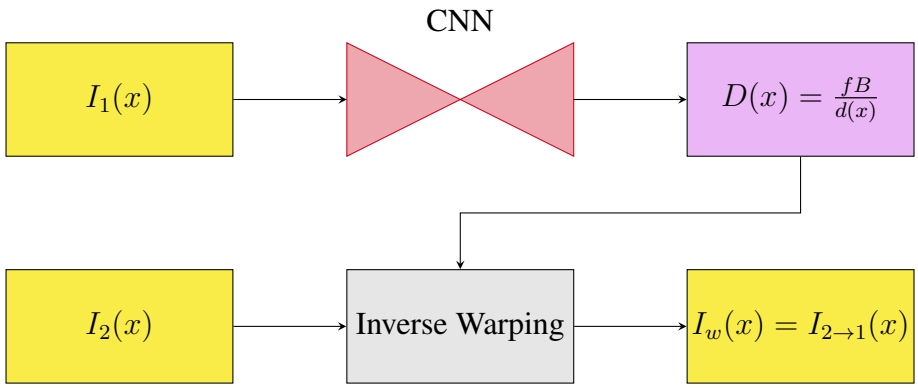

Figure 3. Monocular depth self-supervised Learning. Photometric reprojection error between I_1 and the warped image $I_{2\to1}$. $D(x)$ is the predicted disparity map and is equal to $fB/d(x)$.f is the focal distance, B the baseline between the two cameras and $d(x)$ the depth.

2.2.2 Self-Supervised Learning

The studies presented in this manuscript are mainly derived from self-supervised approaches. This choice stems from the desire to offer an easily adaptable solution that does not depend on the availability of a GT or external sensors. Subsequently it is proposed in this section to deepen the evolution of these approaches and to detail some key points.

It was demonstrated in [30] the capability to predict a monocular depth map using self supervised learning. It is based on minimizing the photometric reprojection error between a pair of stereo calibrated images, as illustrated in Figure 3. The predicted disparity is used to warp the second image (I_2) into the first one (I_1) to minimize the loss $\mathcal{L}_{reproj} = \|I_w - I_1\|^2$, with I_w the reprojection of I_2 into I_1. Yet, the photometric error is non-informative in textureless areas permitting equally good disparity map to coexist. This issue was initially solved by adding a L2 regularization $\mathcal{L}_{smooth} = \|\nabla D\|^2$ to smooth the disparity prediction (D) by penalizing discontinuities. These two functions result in a weighted sum used to optimize the depth prediction as follows:

$$\mathcal{L}_{global} = \mathcal{L}_{reproj} + 0.01 \cdot \mathcal{L}_{smooth} \tag{2.3}$$

The loss function has since been improved [32] by combining a L1 and a single scale SSIM [83] to better optimize the photometric reprojection, in equation 2.4. The two terms are weighted by a constant generally at $\alpha = 0.85$ [32]. Also, the previously used smoothing function induces a loss of sharpness at the edge of the objects in the image. This effect has been corrected by making the smoothing function sensitive to edges, by basing it on the

gradient of the image ∂I, as described in equation 2.5.

$$\mathcal{L}_{reproj} = \alpha \frac{1 - \text{SSIM}(I_1, I_w)}{2} + (1 - \alpha) \|I_1 - I_w\| \tag{2.4}$$

$$\mathcal{L}_{smooth} = |\partial_x D| e^{-\|\partial_x I_1\|} + |\partial_y D| e^{-\|\partial_y I_1\|} \tag{2.5}$$

These two terms still remains the core elements of the self-supervised loss function as of today. However, improvements have been made to increase the accuracy and make training more flexible. We can note the addition of the left-right consistency which proposes to compare the disparity predictions made on the left and right images [32]. Furthermore, by coupling the prediction to a posenet [44], it is possible to obtain the relative pose between two images and thus to work with a sequence instead of a stereo pair [92].

One of the major papers on self-supervised depth prediction, monodepth2 [33], proposed to combine optimization on a sequence and on the stereo pair to combine the best of both worlds. Indeed, this allows to increase the number of viewpoints to refine the optimization and to avoid occlusion problems. The latter is addressed that in case of good prediction, an occluded area induces a higher photometric error than a non-occluded area. Moreover, by multiplying the points of view, it is likely to eliminate all blind spots. Consequently, the minimum reprojection error per pixel is preferred to the average error. This last framework remains a reference in monocular depth prediction as it presented solid precision and published an easy to use code available on github.

New approaches have since succeeded in improving these performances by proposing some adjustments. They include the use of depth maps computed from a stereo pair. It allows to distill this prediction to the monocular network during training with the equation 2.6 [11, 60]. But the distillation is only transmitted on failing cases, i.e. where the photometric error with the monocular prediction is greater than the one computed with the stereo depth. This guarantees not to degrade the prediction by adding this distillation term.

$$\mathcal{L}_{dist} = \log(|D - \hat{D}| + 1) \tag{2.6}$$

Year	Methods	Sup	Param	AbsRel↓	SqRel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
2014	Eigen [25]	S	71M	0.190	1.515	7.156	0.270	0.692	0.899	0.967
2019	Wang [81]	S	-	0.088	0.245	**1.949**	0.127	0.915	0.984	0.996
2020	Patil [59]	S	-	0.102	0.655	4.148	0.172	0.884	0.966	0.987
2021	Bhat [4]	S	78M	**0.058**	**0.190**	2.360	**0.088**	**0.964**	**0.995**	**0.999**
2022	Huynh [43]	S	2.1M	0.133	-	5.157	-	0.842	0.948	0.980
2016	Garg [30]	U	17M	0.152	1.226	5.849	0.246	0.784	0.921	0.967
2017	Zhou [92]	U	42M	0.208	1.768	6.865	0.283	0.678	0.885	0.957
2017	Godard [32]	U	31M	0.133	1.142	5.533	0.230	0.830	0.936	0.970
2018	Poggi [61]	U	1.9M	0.153	1.363	6.030	0.252	0.789	0.918	0.963
2019	Godard [33]	U	15M	0.106	0.818	4.750	0.196	0.874	0.957	0.979
2020	Liu [50]	U	0.2M	0.141	1.080	5.264	0.216	0.825	0.941	0.976
2020	Zhao [91]	U	-	0.113	**0.704**	4.581	0.184	0.871	0.961	**0.984**
2021	Peng [60]	U	27M	0.099	0.754	4.490	0.183	0.888	0.963	0.982
2021	Watson [84]	U	-	**0.098**	0.770	**4.459**	**0.176**	**0.900**	**0.965**	0.983
2022	Tao [76]	U	14M	0.107	0.765	4.532	0.184	0.893	0.963	0.983
2017	Kuznietsov [48]	Semi	81M	0.113	0.741	4.621	0.189	0.862	0.960	0.986
2019	Cho [17]	Semi	-	**0.099**	0.748	5.599	0.183	0.880	0.959	0.983
2020	Guizilini [36]	Semi	-	0.102	0.698	4.381	0.178	**0.896**	0.964	**0.984**
2022	Baek [3]	Semi	18M	0.101	**0.657**	**4.262**	**0.176**	0.892	**0.966**	**0.984**

Table 2. Comparison ot the accuracy of monocular depth methods from the state-of-the-art on the Kitti dataset [31] on the Eigen split [25] benchmark evaluation. Input image size is 192×640 and the output depth range is from 0 to 80m. Methods are separated by supervision type with "S" for supervised, "U" for unsupervised and "Semi" for semi-supervised. Param corresponds to the networks parameters, when available. The best results per type is **bold** and the second best is underlined.

$$\text{Abs Rel}(x, y) = \frac{1}{N} \sum_{i}^{N} \frac{|x_i - y_i|}{y_i} \tag{2.7}$$

$$\text{Sq Rel}(x, y) = \frac{1}{N} \sum_{i}^{N} \frac{(x_i - y_i)^2}{y_i} \tag{2.8}$$

$$\text{RMSE}(x, y) = \sqrt{\frac{1}{N} \sum_{i}^{N} (x_i - y_i)^2} \tag{2.9}$$

$$\text{RMSE Log}(x, y) = \sqrt{\frac{1}{N} \sum_{i}^{N} \left(\log(x_i) - \log(y_i)\right)^2} \tag{2.10}$$

$$\text{a1}(x, y) = \frac{1}{N} \sum_{i}^{N} \left(\max(\frac{x_i}{y_i}, \frac{y_i}{x_i}) < 1.25 \right) \tag{2.11}$$

$$\text{a2}(x, y) = \frac{1}{N} \sum_{i}^{N} \left(\max(\frac{x_i}{y_i}, \frac{y_i}{x_i}) < 1.25^2 \right) \tag{2.12}$$

$$\text{a3}(x, y) = \frac{1}{N} \sum_{i}^{N} \left(\max(\frac{x_i}{y_i}, \frac{y_i}{x_i}) < 1.25^3 \right) \tag{2.13}$$

2.2.3 Evaluation Metrics

The depth prediction quality is usually measured with comparison to GT depth. Yet, as mentioned earlier, the GT can be difficult to acquire, thus a restricted number of datasets can be used for the evaluation. In addition, it is better to compare with other state-of-the-art methods, so the same datasets are often used. Indeed, the performances of a network is linked to its training set and may produce poor results when tested outside its domain of expertise. For a fair evaluation, the methods must have been trained on the same dataset. Subsequently, Kitti [31] is the most commonly used dataset for monocular depth evaluation, even though it has been released 10 years ago. But this choice is not only made by default. It is an easily accessible dataset, offering numerous urban navigation sequences and equipped with a calibrated sensor set including cameras, IMU and LiDAR. Alternatively, the NYU-Depth V2 dataset [57] is largely used for indoor applications.

The method performances are measured with a pixel-wise evaluation on the following error functions. They are respectively the absolute relative, the square relative, the root mean square error (RMSE) and the RMSE on logarithmic values. The last three equations a1 to a3 are the accuracy measuring of the likelihood between the prediction and the groundtruth at three different threshold levels. The benchmark remains the same since the beginning and is usually called Eigen split [25] in reference to the author. It is a set of 697 test images on which are aligned the GT corresponding to the LiDAR point cloud at the timestamp.

In table 2 are gathered performances of monocular depth prediction methods from the state of the art. As anticipated, supervised methods are achieving higher accuracy as GT is available during training. As such, Bhat et al. [4] are proposing to use transformers architecture for a more adaptative prediction, resulting in higher precision but at the cost of a bigger network with 78M parameters. In constrast, the semi-supervised training does not seem to produce much better results than self-supervised. However, it can be noticed the state-of-the-art keeps being regularly outperformed. It is therefore likely that the performance will be further improved in the coming years.

There is also a great disparity in the number of network architecture parameters. These data are unfortunately not always available, which complicates the analysis. However, the more complex architectures tend to obtain better results. Indeed, more operations are performed which allows a great flexibility inducing a better prediction to some extent. Yet, more parameters generally implies a higher network complexity, requiring more training with more data and result in a slower inference runtime. Subsequently, some approaches are proposing lightweight architecture for real-time monocular depth estimation [43, 61, 50].

2.3 Semantic Segmentation

Image segmentation is an essential task in computer vision and image processing, as it allows to add understanding in a scene. It is therefore widely studied and implemented in all domains whether it is autonomous navigation, medical imaging, video surveillance and many others. A wide variety of algorithms have been developed to solve this problem, including thresholding [46], k-means clustering [37] or Markov random fields [21]. However, as in many other fields, these methods have been dethroned by deep learning applications due to their game changing performances. They managed to get the best results on all popular benchmarks and exceeded all expectations in the field [55]. Therefore, this section covers the deep learning aspect of the semantic segmentation, from an overview of some notable architectures, to the main datasets employed and the limits and expectations of the approach.

2.3.1 Deep Learning Architectures

Most of the improvements in prediction performance for semantic segmentation are due to the development of new network architectures. Indeed, the training procedure is essentially the same from one method to another and relies primarily on the per-pixel minimization of the cross-entropy loss function. The method is trained to reproduce the segmentation classes fixed by the ground truth data. This section presents the principal methods that have marked the evolution of semantic segmentation prediction.

One of the first semantic segmentation approaches was proposed in [52]. It is a fully convolutional network (FCN) composed, as the name suggests, only of convolution layers while removing the fully connected layers. It provides the advantage of working with different image sizes, making it easily adaptable to the input data. It outperformed the methods of the time and demonstrated the capability to train a network to learn segmentation. Yet, it has the downside of being too slow for realtime applications and is not able to consider global context of the image. The latter point has been corrected in ParseNet [51] by adding a context vector into the prediction process consisting of the averaged feature maps.

Eventually, the FCN has been put aside and replaced by CNN architectures. These approaches remain widely popular due to their versatility and performances. The most popular model is undoubtedly the encoder-decoder. The encoder performs a series of convolutions, nonlinear activation functions and pooling operations to extract a low-dimensional feature map, called latent space, from the input image. This latent space

is then decoded via a series of transposed operations, similar to those performed in the encoder to recreate an output image containing the inferred data.

CNN methods were initially based on VGGNet [72], which were designed by investigating the effect of the convolutional network depth on the accuracy. It resulted in a simple, yet efficient, architecture composed of a set of convolution layers, pooling layers and ended up with fully connected layers. Badrinarayanan et al. proposed in SegNet [2] an adaptation of VGGNet by removing fully connected layers to make the network lighter. This new architecture has made it possible to divide the number of parameters by ten while remaining more efficient.

The U-Net [64] architecture, originally designed for biomedical image segmentation, proposed a solution to reduce the number of images required for training and to localize high resolution features. It is achieved by feeding into the decoder feature maps predicted after each layer of the encoder. It permits to contract the network to better capture the image context and enable features localization. Indeed, the reinjection of features from the encoder permits gradient decay during backpropagation inducing to loss pattern information. From this architecture emerged many others and its structure is still widely used, especially in other fields, such as for depth prediction [92].

As demonstrated by the previous methods, the key to improve prediction accuracy lies in two principles: the network depth and the training data. These two aspects tide together, as a deeper network requires more images to be correctly trained. Indeed, it is composed of more parameters, i.e. more variables to optimize. This problem has been addressed by ResNet [39] which proposed a residual learning framework. It consists of combining together the input and the output of weighted layers sequence, while allowing to increase the depth of the network without changing the number of parameters. It is therefore easier to train while resulting in improved results. The simplicity and efficiency of the framework makes it still highly used on methods with the best performances [8, 27].

To limit the network complexity, CNN like VGGNet or ResNet are primarily composed of convolutions with kernel size of 3×3. To extend this receptive field, Dilated Convolutional Models introduced a dilatation rate to transform a 3×3 kernel into 5×5 while preserving its 9 parameters. This framework has been popularized with the Deeplab family [7, 8], which presents increasingly good results.

Most recently, segmentation tasks have explored the use of attention-based methods [28]. They permit to weight each individual pixel to grant a variable level of importance to areas in the image. This attention mechanism has the advantage of being more efficient than the

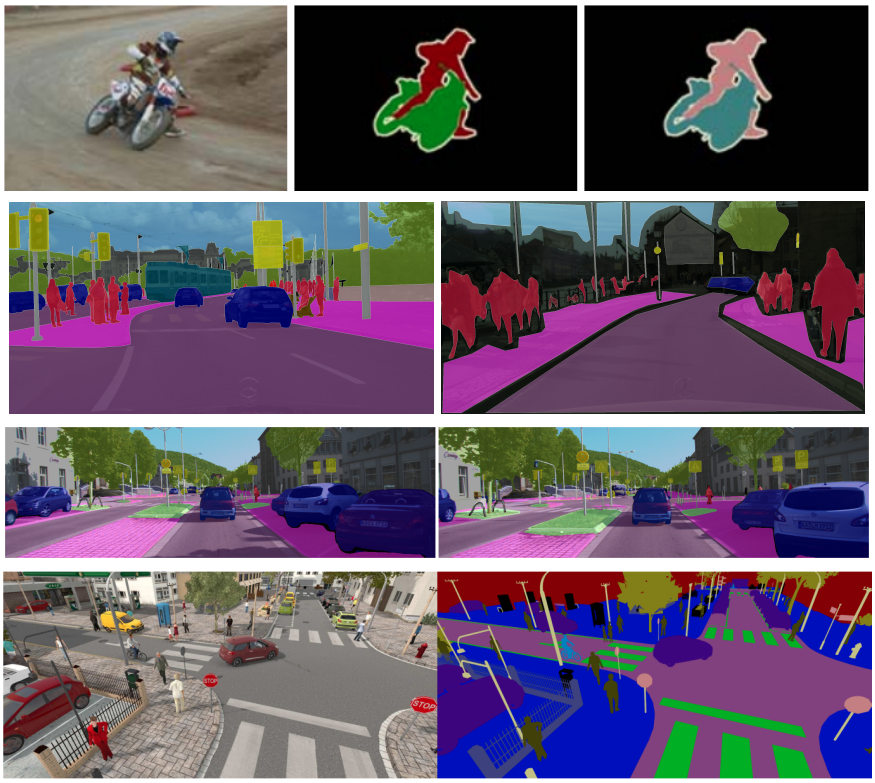

Figure 4. Overview of popular datasets for image segmentation. From top to bottom, PASCAL Visual Object Classes (VOC) [26], Cityscapes [20] with its fine and coarse segmentation, Kitti-STEP [85] and SYNTHIA [65].

classical pooling operations [55].

2.3.2 Segmentation Datasets

It is proposed in this section to present the main datasets used to train and evaluate segmentation methods, some of which are illustrated in Figure 4. Segmentation approaches are nowadays mostly supervised, which means that each of these datasets must propose labels for each of their classes. As the creation of annotations is costly, the number of images with GT is necessarily limited. As explained previously, the number of available data will condition the training capacities of a network and thus the accuracy of its prediction. To overcome this problem, it is possible to use an augmentation process. It consists in applying some transformations on the images to generate new ones which will be perceived differently by the network. These transformations can take various forms such as a homographic transformation of the image or a modification of the feature map or a combination of both. This allows to increase the performance of the network on small datasets, by limiting overfitting.

The PASCAL Visual Object Classes (VOC) [26] is a highly popular dataset as it proposes

annotations for classification, segmentation, detection, action recognition and person layout. It consists of a set of over 40,000 annotated images into 20 classes representing objects, animals or persons, thus the background is not considered. It also defined the intersection over union (IoU) function, which remains the most commonly used metric for semantic evaluation:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{TN} + \text{FN}} \tag{2.14}$$

where TP, TN and FN are respectively the true positives, the true negatives and the false negatives.

The Cityscapes [20] dataset has been specifically designed for urban scene semantic understanding. It presents a key challenge for autonomous navigation applications as it offers a variety of street scenes acquired from 50 different cities. It consists of a set of 5,000 annotated images with an addition of 20,000 coarsely annotated images as illustrated in Figure 4. It contains 25 labeled classes, themselves organized into 7 groups.

The Kitti [31] dataset was not published with any annotated data for image segmentation. However, due to its great popularity for navigation applications, many people have offered their own annotations. A segmentation dataset Kitti-STEP [85] has recently been published, proposing 20,000 annotated images. The data are based on the Cityscapes classes and were generated using a DeepLab based network [12]. The predicted segmentations have then been refined by human annotators.

As the number of available datasets proposing labeled images are limited, the usage of simulators such as SYNTHIA [65] permits to generate realistic synthetic data. It aims at being combined with existing real world annotated data to improve performances. Similarly, SynWoodscape [69] dataset aims at simulating the Woodscape [88] dataset by recreating similar conditions. It permits to largely expand the annotated data which were limited on the real-world dataset.

2.3.3 Perspectives

The need for access to a large amount of annotated data for training is an obstacle for segmentation prediction. Subsequently, weakly supervised and unsupervised methods are starting to appear. Therefore, transfer learning based approaches [71] are being developed. It consists in pretraining a network on a large dataset to predict generic classes and to use it

to fine-tune another network on a few images to learn a more specific segmentation. Fully self-supervised training presents the next challenge in semantic segmentation. However, this raises many questions on how to achieve this segmentation and certainly requires rethinking the way to train these networks.

As for depth prediction, the main criterion demonstrating the quality of a method is the accuracy of its prediction. The execution time is therefore secondary and rarely measured. Many approaches, such as FCN based methods, do not allow real time, with an execution time higher than 100ms. Fortunately, methods like Deeplab, based on the use of dilated convolutions, allow to reduce the execution time by approaching a real-time operation. This means progress can still be made to propose reliable real-time methods.

In the same way as for the execution time, segmenting architectures are generally heavy and require a large amount of GPU memory. However, it can be noted that the trend is to decrease this by proposing lighter innovative architectures. This allows to reduce the number of parameters and thus the need for training data while making the architecture embeddable and working in real time. While lighter architecture struggle to compete with larger architecture, they offer a more favorable ratio between the precision and the memory usage.

Chapter 3

Quadtree Segmentation for Obstacle Avoidance

This chapter presents our contributions to generate a quadtree segmentation for obstacle avoidance. It is based on the Quadtree Generating Networks (QGN) architecture [14], permitting to directly infer a quadtree data structure from an input image. In order to develop an easily adaptable method, segmentation labels are constructed automatically from depth information instead of being annotated by human supervision. After introducing the concept of segmentation for obstacle avoidance, this chapter will deeply describe the QGN used to fulfill the application. Extended experiments will end the chapter to discuss on the interests of the method with respect to a dense equivalent.

3.1 Segmentation for Obstacle Avoidance

This initial section defines the objectives of the method and how the reference segmentation maps are generated.

3.1.1 Objectives

Autonomous navigation is one of the primary subjects in computer vision and robotics. It is a mandatory first step for the system to be able to move safely in its environment in order to achieve any desired tasks. Safe navigation requires for the robot to be able to localize itself in its environment and to avoid obstacles. To accomplish those navigation tasks, systems are typically equipped with sensors able to acquire depth information, like a LiDAR or a RGBD camera. In recent years, deep learning approaches emerged, permitting

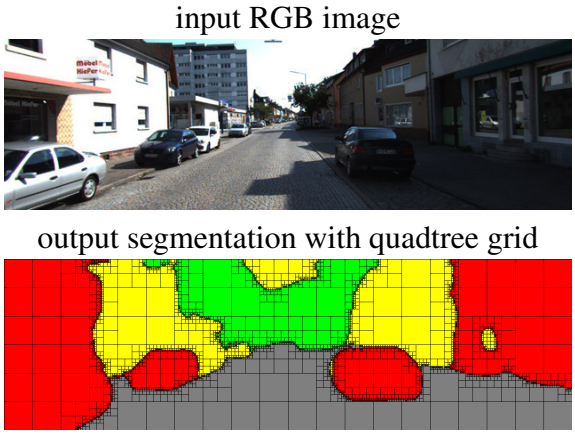

Figure 5. Quadtree scene segmentation for obstacle avoidance.

to infer depth using a single camera as input [25]. As of today, those methods are not as accurate as stereo cameras methods. Yet, they have the advantage of working with a single RGB camera, which avoids the need for a system of synchronized and calibrated cameras, while being more compact.

In this chapter, we focused our interest on obstacle detection from a single image to be used in real-time. Therefore, we proposed a solution using a single camera as input to generate segmentation of the image to support the decision on avoiding obstacles as illustrated in Figure 5. The method uses a deep neural network to predict the segmentation, which will be learnt through a training procedure supervised by a depth aware method. In order the accelerate the inference, the approach is based on QGN [14]. The quadtree decoder presents the advantage of considerably reducing the computational complexity of the network. Indeed, some areas in the image can be classified at low resolution. Thus, the pixels that have already been classified can be ignored in the rest of the decoder, thanks to the usage of submanifold sparse convolutions [9]. It permits to focus the attention on areas requiring a higher degree of accuracy to be segmented.

3.1.2 Generating Labels from Depth Information

3.1.2.1 Depth Segmentation versus Semantic Segmentation

Segmentation networks are broadly used for semantic scene understanding. In the case of navigation, it consists of detecting cars, roads, buildings, and others. Those methods have been largely studied and are able to provide accurate segmentation. Yet, they have with the issue of requiring access to ground truth labels for training. As the semantic information is based on the human understanding of the scene, the labels annotation have to be supervised, which is costly and time consuming. In addition, the quality of the annotations will depend

on who is generating them, inducing bias.

The obstacle avoidance task is generally performed based on depth information as it provides an accurate understanding of the scene's geometry. From the knowledge of the distance to objects, the best path can be derived to navigate safely. The emergence of monocular depth estimation methods have shown that this information can be acquired from a single image. Subsequently, it is proposed to do scene segmentation for obstacle avoidance using a monocular camera. It is initially constructed geometrically from a depth map and be used as ground truth during the training phase.

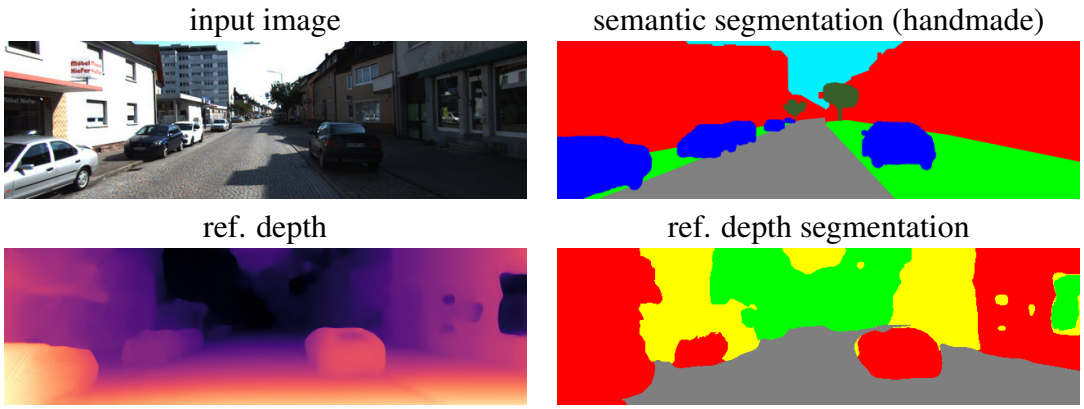

Figure 6. Comparison between semantic segmentation and depth segmentation.

The choice of segmentation classes is made by taking into account the obstacle avoidance objectives. The framework must be able to identify the area where it is safe to navigate as well as to classify the obstacles according to their level of depth. This led to propose a segmentation in four classes: one is segmenting the ground, and the other three are splitting the depth according to the distance of the obstacles. The ground class represents the safe area in which the autonomous system is allowed to navigate. The following three classes are based on the depth and indicate the level of priority with which the system must take into account the obstacles. The proposed segmentation is presented in Figure 6 along the semantic segmentation and the depth map.

This choice of solution does not apprear to be straight forward and could appear counter-intuitive as depth information seems to be a better fit. Yet, by combining the scene segmentation into a limited number of classes and with the usage of Quadtree Generating Network [14], it is possible de generate a light solution proposing complementary results as depth.

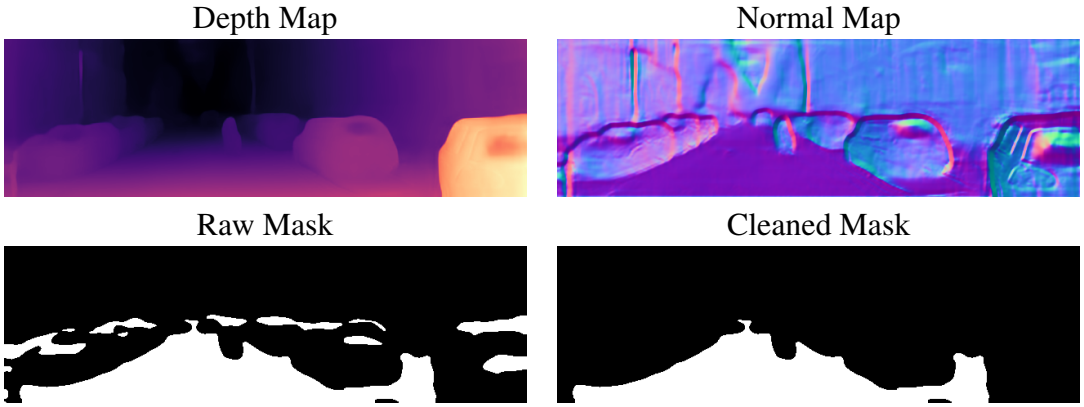

Figure 7. Extracting the ground class from the normal map computed from the depth map. The resulting mask is cleaned to remove the outliers caused by the top of the cars' surfaces.

3.1.2.2 Differentiating the Ground as a Safe Area to Navigate

For urban applications as proposed in the Kitti dataset [31], the viewpoint of the cameras installed on the car is consistent on the entire sequences. It is therefore possible to consistently extract relevant information with basic image processing tools. Indeed, the scene is constantly observed from the same angle. Therefore, the direction of the normal vectors of the ground will remain the same on the entire sequence. The normal field can be computed from the depth map to extract the ground. It corresponds to the safe area where the mobile system is allowed to safely navigate.

$$n(x,y) = \frac{v(x,y)}{\|v(x,y)\|_2}, \text{ where } v(x,y) = [-\frac{\partial z(x,y)}{\partial x}, -\frac{\partial z(x,y)}{\partial y}, 1] \tag{3.1}$$

The normal vector is perpendicular to the surface and can be computed from the depth map as described in Equation 3.1. The depth value is noted $z(x,y)$ in the point (x,y). The direction of the normal vector can be obtained by computing the local derivative of the depth in the x and y direction. It can be achieved in practice by computing the image gradient using a Sobel filter as illustrated in Figure 7. By looking at the normal map, it is visible that the ground can be extracted by a simple threshold. Yet, it results with having some outliers, mainly corresponding to the top of the cars, which have there normal vectors with the similar direction as the ground as illustrated on the normal map in the Figure 7. With the assumption that the ground represents a large and continuous area; the outliers can be removed by using post-processing computer vision tools to remove small isolated areas. Some outliers may persist in limited numbers such as they will have no affect on the training procedure.

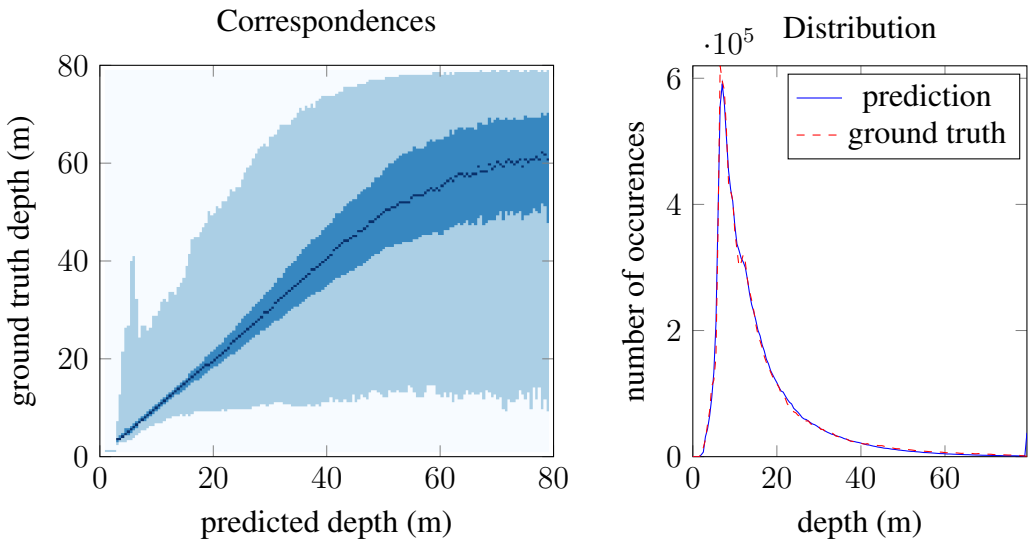

Figure 8. Monocular depth values distribution and correspondences with the ground truth on urban scenes.

3.1.2.3 Applying a Threshold on Depth Values

The segmentation is dividing the images into four classes: one is used to detect the ground, and the other three are classifying the information according to their distance. In this section will be discussed the choice leading to the three classes to represent the depth.

The segmentation will be performed using information from a single image and will be closely related to depth. Therefore, the segmentation into classes must be made with knowledge of the general behavior of monocular depth network performances on the depth prediction. Indeed, even if our method does not directly predict depth, it is highly related to it and is expected to replicate similar behavior. This analysis is done by extracting the data of the testing set from the Kitti dataset [31]. It is composed of original images not used during the training procedure, with access to ground truth depth data acquired using a LiDAR sensor. The objective of this study is to visualize how is predicted the depth by searching for correspondences and distribution with respect to the ground truth, as illustrated in Figure 8.

The correspondence graph consists of doing a pixel-matching between the predicted values and the corresponding ground truth depth values. The graph is representing the disparity of the prediction as box plot. The darker blue is the median, and the blue represents the values contained between the first to the third quartile and the light blue covers the lower to the higher limit, which represents 99.65% predicted values. The second graph represents the data distribution per depth value for the prediction and ground truth. The two curves largely overlap as the data distribution is highly similar.

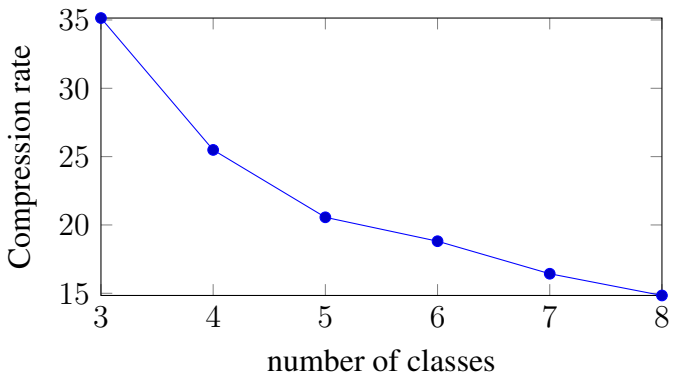

Figure 9. Image compression rate due to the quadtree representation as a function of the number of segmentation classes.

These two graphs are highlighting the nature of the urban depth information, with a significant majority of the information extracted from the images representing depth below 20 meters. These over-representations can be seen on the correspondence graph on the left, where the error is much smaller below 20 meters. Indeed, the network was able to optimize the quality of its prediction on this interval during the training. It results in a trustworthy depth inference at short range and a hazardous one at long range, beyond 40 meters.

The first intention was to segment the depth into two classes: the close obstacles and the rest which can be ignored. It would have resulted in an image segmented into three classes, permitting a high compression rate of the image with the quadtree representation, as illustrated in Figure 9. However, by analyzing the monocular depth prediction, it seemed patent that the depth splitting into two classes will lack reliability. Indeed, it would have result with a sizeable percentage of false positives. Adding more classes inevitably result in the decrease of the image compression rate, as each class will describe smaller areas. Based on Figure 9, the decision is made to limit the prediction to four classes to keep a high compression rate while still obtaining a reliable prediction. Indeed, separating the depth values into three classes offers the advantage of having a middle class acting as a buffer between the close obstacles to cautiously consider and the far obstacles that can be ignored.

3.1.3 Self-supervised Reference Map

The previous sections presented the solutions to extract the different classes from a depth map. In this section is presented the implemented algorithm to infer the dense depth information. Except for the specific case of the monocular depth estimation, at least two images are required to triangulate the position of the obstacle and estimate the depth. For cost efficiency, depth is generally acquired from a calibrated stereo system. Standard

Method	Input	Size	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
Ours	Stereo	192 x 640	0.072	0.681	3.924	0.168	0.931	0.965	0.981
EPCDepth [60]	Mono	192 x 640	0.099	0.754	4.490	0.183	0.888	0.963	0.982
SingleNet [11]	Mono	320 x 1024	0.094	0.681	4.392	0.185	0.892	0.962	0.981
StereoNet-D [11]	Stereo	320 x 1024	0.048	0.482	3.393	0.105	0.969	0.989	0.994

Table 3. Quantitative results on the Eigen split evaluation set [24].

computer vision algorithms can be used to perform stereo matching to compute the pixels disparity between the two images. As the system is calibrated, accurate depth, noted d, can be obtained from the computed disparity, noted D. With knowledge of the baseline between the two cameras and the focal length respectively noted b and f, in Equation 3.2, for every pixel p_i in the image.

$$d(p_i) = \frac{b \cdot f}{D(p_i)} \tag{3.2}$$

However, those standard computer vision methods tend to compute a sparse depth map due to the occlusion in the images between views. Convolutional neural network depth prediction methods have demonstrated their capability to efficiently fill the gaps, by hallucinating what might represent the correct depth in the occluded areas. Some approaches, such as PSMNet [6], are proposing efficient solution for stereo matching depth estimation but results in a heavy network using 3D convolutions blocks. A lighter architecture has been preferred based on monocular depth networks [11, 60] but with a pair of images as input. This network has been trained using state-of-the-art monocular depth learning methods on the Kitti dataset [31]. It has accuracy results in between the monocular and the stereo performances as illustrated in table 3. It could have been possible to adopt a method proposing higher accuracy, such as StereoNet-D [11], but the weights and the implementation of the network are not openly accessible. Therefore, it was preferred a method implementing a simpler architecture, but with a completely known and controlled pipeline.

The overview of the complete reference segmentation process is presented in Figure 10. The depth is predicted with our stereo network and used to generate the segmentation. The ground class is extracted from the normal map and applied on top of the three depth related classes. It means it has the priority and is substituted for the other classes.

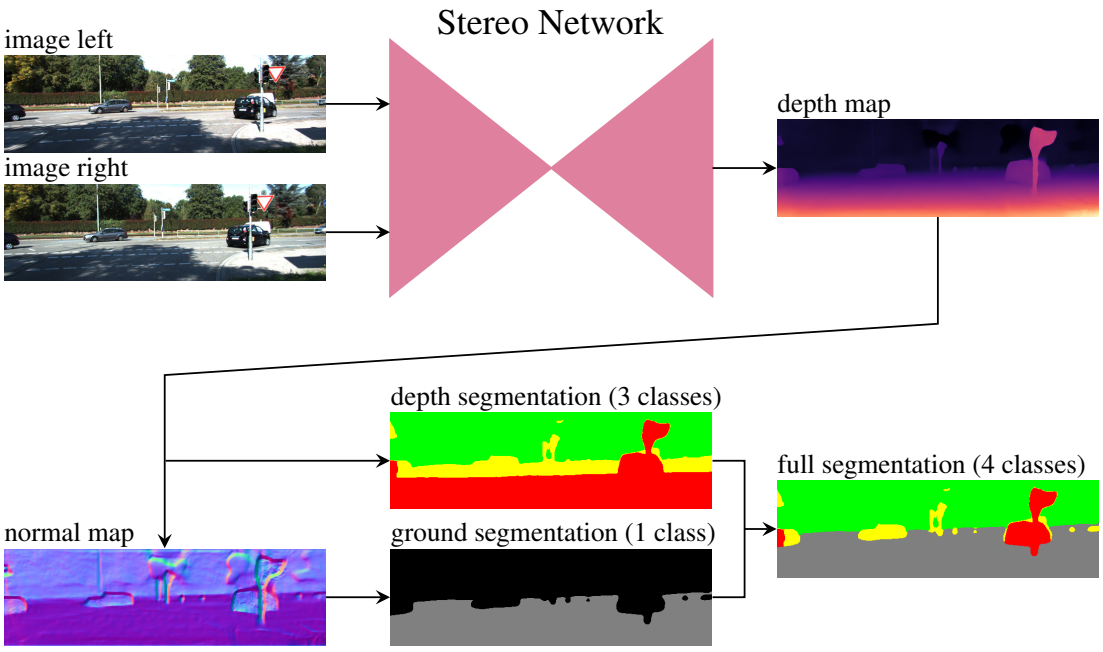

Figure 10. Overview of the full segmentation computation, used as reference to train our network.

3.2 Quadtree Generating Networks for Segmentation

This second section now focuses on the implementation of the method by presenting the architecture of the network, the specificities of the quadtree prediction and the training phase.

3.2.1 Network Architecture

The network illustrated in Figure 11 is based on the quadtree generating network (QGN) proposed by [14]. It is a U-Net [64] architecture composed of a dense ResNet [39] encoder and sparse ResNet decoder. The sparse decoder consists of submanifolds sparse convolutions blocks [35], which exclusively performs operations on active sites, i.e. where there is data to be processed. Initially designed for sparse input data processing, they are suited for quadtree construction, where the information is voluntarily reduced.

The strategy behind QGN is to directly generate a quadtree through the decoder without having to compute the maximum resolution dense prediction. Indeed, some information can be correctly predicted at low resolution from the first layers of the decoder. Further processing of this information will not amend the output. Subsequently, they can be stored into the quadtree and removed from the set of active sites to be unprocessed by the following layers of the decoder.

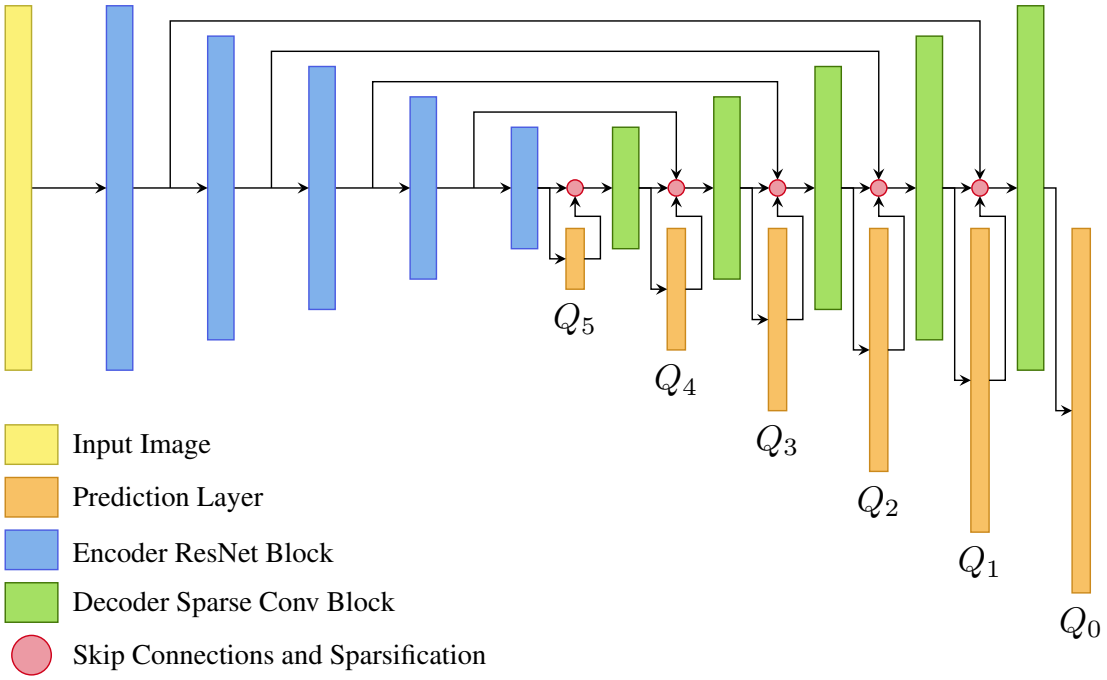

Figure 11. Architecture of the quadtree segmentation network. It is based on a U-Net with a dense ResNet encoder in blue and a sparse decoder in green. The prediction layers, in orange, are generating each level of the quadtree from low to high resolution.

The active sites represent the set of features that needs to be processed by the sparse convolution. It can be considered as a binary mask applied to the tensor where the active sites are the values associated with a 1 in the mask. Thus, it makes it possible to modify the set of active sites as desired to decide what to be processed in the subsequent layer of the decoder. This sparsification process is done at the same time as the skip connection and is represented by the red circles in Figure 11. The set of actives sites is decided based on the segmentation obtained at the output of the prediction layers.

3.2.2 The Mixed Class

Each prediction layer in the decoder outputs a sparse segmented map, noted Q_l with l representing the quadtree level, as presented in Figure 11. The activation masks can be built from the segmentation, which will define the set of active sites for the subsequent layer of the decoder. It is obtained by applying a filter on the values of Q_l to determine which class obtained the highest score. In standard segmentation methods, the highest score indicates to which class a pixel belong. For quadtree segmentation, the map is constructed recursively from low to high resolution. Therefore, it is acknowledged that areas in the image can be composed of several classes, thus have to be subdivided to be classified. It is represented by the introduction of a *mixed* class, as proposed in [14].

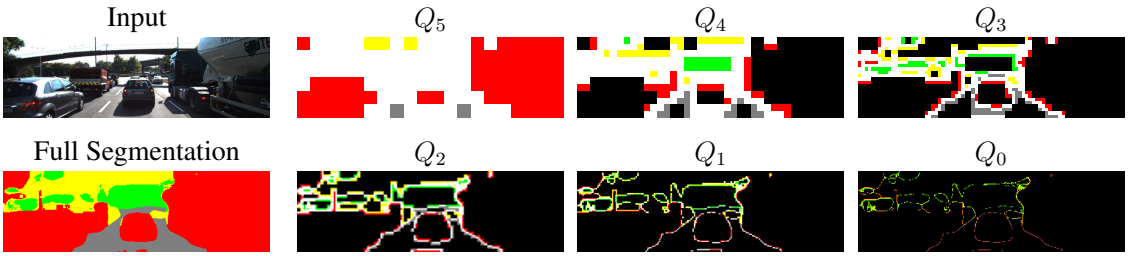

Figure 12. Quadtree prediction decomposition. On the top left the input image and the bottom left the recomposed output prediction. Q_5 to Q_0 are the outputs of each prediction layers. We have the *mixed* class in white, the *Ground* class in gray, the *Close* class in red, the *Middle* class in yellow and the *Far* class in green. The pixels in black stores no value: it's the area which have been predicted by previous layers of the decoder.

The addition of a *mixed* class permits to indicate the area that cannot be classified at the current resolution. Therefore, the corresponding pixels have to be subdivided to seek more details. Regarding the other pixels that have been successfully classified, it is unnecessary to look for more details at higher resolution. Consequently, they can be withdrawn from the active sites and will not be processed in the subsequent layers of the decoder. The activation map $\mathcal{A}$ is defined for each value v_i as follow:

$$\mathcal{A}(v_i) = \begin{cases} 1, & \textit{if } \arg\max(v_i) = m, \\ 0, & \textit{else,} \end{cases} \tag{3.3}$$

where m is the channel number of the *mixed* class, and i is the index of the pixel in the segmentation map.

3.2.3 Training

3.2.3.1 Loss Function

As in [14], the learning process is based on the multi-scale minimization of the cross-entropy loss function, noted $\mathcal{H}$. The optimization is performed between each of the predictions Q_l and the reference map decomposed into quadtree Q_l^* for every classes on every scale level composing the quadtree. In our study case, the quadtree is composed of six levels, inducing $l = \{0, .., 5\}$. The loss is computed for each level of the quadtree Q_l as follows:

$$\mathcal{L}_l = \frac{1}{N_l} \sum_{i=1}^{N_l} \mathcal{H}(v_i, Q_l^*(x_i, y_i)) \tag{3.4}$$

where N_l is the number of elements in Q_l. The function $\mathcal{H}$ compares the predicted value v_i to the reference value at the same location in the image (x_i, y_i), such as $Q_l^*(x_i, y_i) = v_i^*$.

Considering the Q_l are sparse predictions, there is a portion of the image that is not evaluated as it does not store any prediction value. As illustrated in Figure 12, this portion represented in black, is growing bigger after each prediction, with the latest only representing a fraction of the image. For them in order to remain impactful in the minimization process, the global loss is a weighted sum of the $\mathcal{L}_l$ terms as presented in Equation 3.5, with λ_l a coefficient defined empirically before the training.

$$\mathcal{L} = \frac{1}{6} \sum_{l=0}^{5} \lambda_l \mathcal{L}_l. \tag{3.5}$$

3.2.3.2 Guided Supervision

The use of QGN offers the advantage of only computing the necessary operations to predict the quadtree. The drawback is the prediction quality depends on the capability of the network to correctly identify those operations. As explained in section 3.2.2, this decision is made based on the set of active sites which indicates to the sparse convolution the features to consider. During the inference, these active sites are defined by the prediction from the previous layer, at the locations where the *mixed* class gets the highest score. This behavior can cause some issues in the early stage of the training, where the network lacks of efficiency.

With a hazardous prediction, it is certain that the quadtree structure will be completely different from the expected one. As explained earlier, the optimization procedure is only able to evaluate the predicted values. With a completely unpredictable quadtree structure, it is impossible to estimate in which direction the optimization will converge. Even in a case of a convex optimization with a unique solution, supervised by the reference map, it is impossible to foresight the time required for a proper training. To address this issue, it is proposed to guide the training by providing to the network the proper subdivision as defined by the reference map, regardless of the network's prediction. The differences between the guided and not guided training are presented in Figure 13 and clearly illustrate the guidance allowing for better optimization.

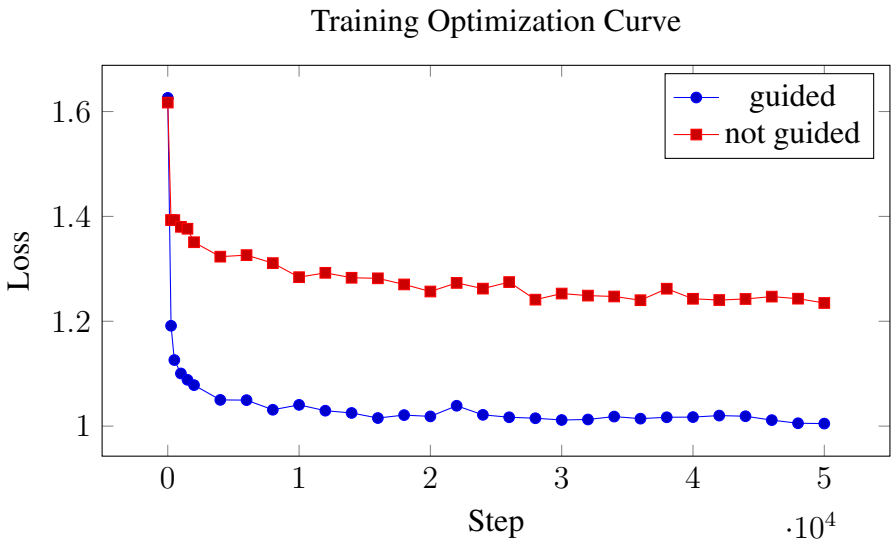

Figure 13. Training optimization curve. Comparison of the Loss value between the guided and not guided training over the 50,000 first steps of the learning.

3.3 Experiments

Experiments are conducted on the Kitti dataset [31]. It offers urban images acquired from cameras in a car, which allows exploring the scene from a constant point of view on entire video sequences. Our method is evaluated under two architectures: sparse ResNet 18 and 50 and compared with the equivalent dense version, which has been trained using the same loss function but to output a dense prediction.The double objectives of those experiments are to evaluate the reliability of the proposed segmentation and the capability of the sparse methods to equal or outperform its dense equivalent.

3.3.1 Qualitative Results

Qualitative results are presented in Figure 14, allowing a visual comparison of the methods with the reference map. From a global observation, all methods are providing an equivalent quality of prediction. It is an anticipated result considering they all followed the same training procedure, with having differences only in their architecture. It also implies that the use of the QGN algorithms allows results equivalent to those of standard CNN methods. Yet, sparse ResNet 18 method, using QGN, seems to be noisier in its detection of *close* obstacles represented by the red class. The biggest differences between methods are at the junction between the *middle* and *far* range classes, respectively in yellow and green. They are both depth related classes, and are describing the extended range of scenes, which can be inaccurate from a single image, as previously illustrated in Figure 8.

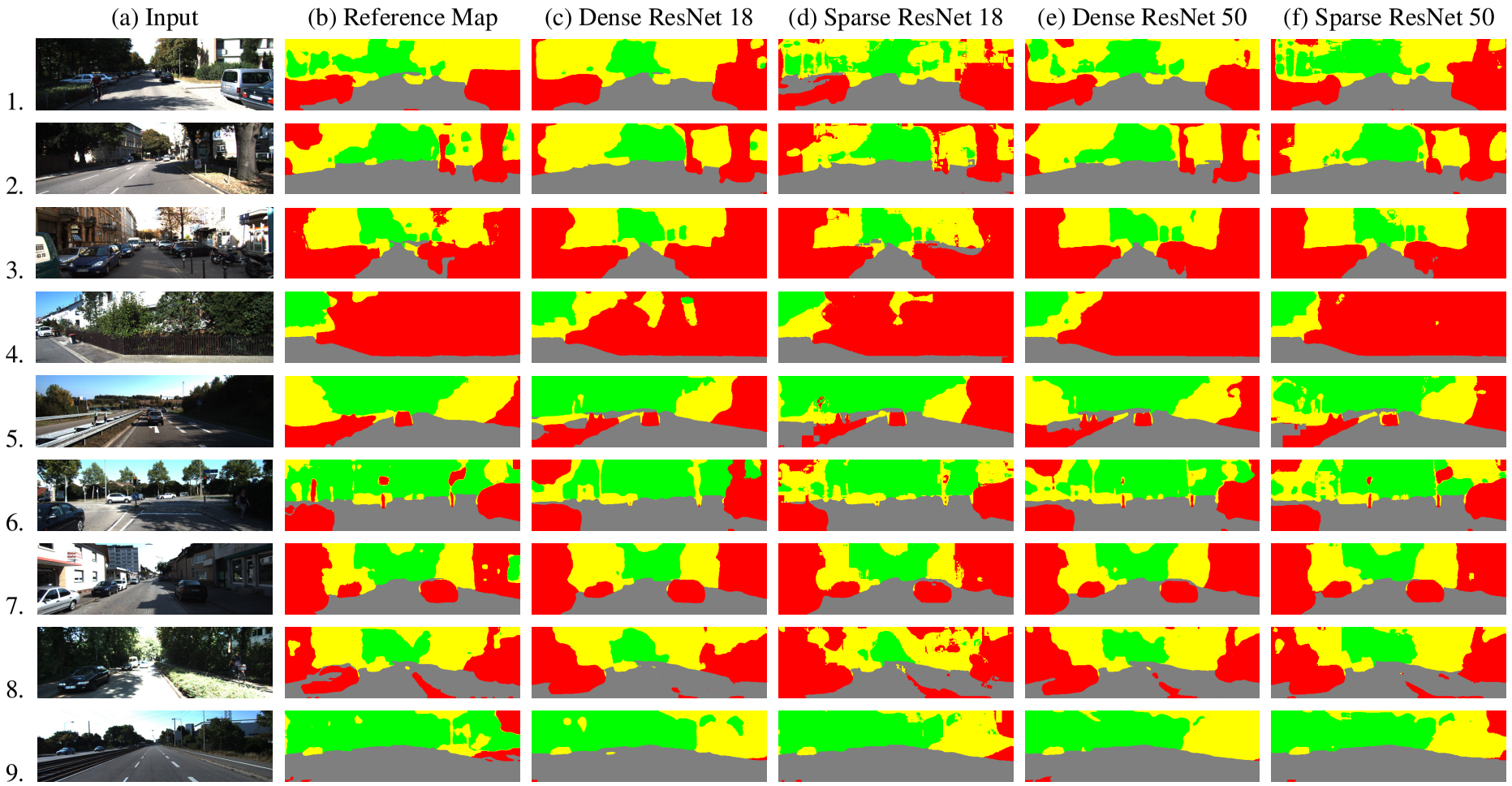

Figure 14. Qualitative prediction overview. From left to right, the single input image (a), the reference map (b) constructed in self supervised manner from a depth prediction network segmented into classes, then from (c) to (f) are the prediction of each methods.

Methods	Close (red)	Middle (yellow)	Far (green)	Ground (gray)	mIoU
Dense ResNet 18	68.48%	54.8%	66.5%	90.7%	70.1%
Sparse ResNet 18	65.3%	50.1%	65.2%	89.9%	67.6%
Dense ResNet 50	**72.1%**	**60.9%**	68.8%	90.2%	73.0%
Sparse ResNet 50	71.9%	59.4%	**70.6%**	**91.4%**	**73.3%**

Table 4. Intersection over Union per class and in average (mIoU).

This sample of nine images from the Kitti testing set provides a broad variety of the method behavior. It is capable to detect close obstacles ahead as in 4^{th} image or even no obstacle at all in the 9^{th}. It also highlights the limits of using a self-supervised method as a reference instead of a certified ground truth. Indeed, in the 7^{th} image, the store's window on the right is classified as a *far* obstacle in the reference map. Similarly on the 8^{th} image, the car on the left is partially classified as *ground*. As presented earlier, the reference segmentation is constructed by transforming depth data, without having a global understanding of the scene. Still, the segmentation networks managed to correct those limitations to provide a more accurate representation than the one from reference. Furthermore, it can also be observed that Sparse ResNet 18 generates a coarser prediction. In images 2 and 8, the segmentation between the *close* and *middle* classes is uncertain in some areas, leading to speckled results. This behavior is not desired, as it necessarily increases the size of the quadtree by forcing access to high resolution pixel-level details.

3.3.2 Accuracy

The table 4 compares the accuracy of the methods on the intersection-over-union (IoU) criterion for each class and on average (mIoU). This metric evaluates the capability of the method to correctly classify the information in the image and can be computed by applying the following equation:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.6}$$

The first observation is the similarity in the results between the sparse and dense methods, with an advantage for the sparse ResNet 50. The *ground* class obtains in all cases the most accurate prediction. The method seems to have difficulties to correctly classify the buffer *middle* class, which is in between *close* and *far* obstacles. As explained in section 3.1.2.3, this result was expected due to the edge effects between the depth-based classes.

Dense 18 / Sparse 18	Dense 50 / Sparse 50	Ground Truth							
		Close		Middle		Far		Ground	
Prediction	Close	**90.18%**	**91.09%**	7.03%	6.47%	0.34%	0.23%	2.45%	2.21%
		88.38%	**91.22%**	8.12%	6.20%	0.47%	0.25%	3.04%	2.33%
	Middle	16.89%	12.69%	**73.51%**	**79.28%**	8.49%	6.97%	1.12%	1.06%
		21.79%	13.84%	**67.40%**	**77.55%**	9.24%	7.35%	1.56%	1.26%
	Far	2.17%	1.01%	22.07%	19.08%	**75.24%**	**79.39%**	0.53%	0.53%
		2.22%	1.02%	21.47%	19.22%	**75.52%**	**79.06%**	0.79%	0.69%
	Ground	3.11%	2.55%	1.26%	1.11%	0.57%	0.54%	**95.06%**	**95.79%**
		2.98%	2.53%	1.17%	1.08%	0.49%	0.51%	**95.36%**	**95.88%**

Table 5. Confusion Matrix representing the segmentation distribution per class of the prediction with respect to the ground truth for each method. All four are presented in the same table with their values exposed in the corners of each cell in the order indicated at the top left of the table. Green **bold values** on the diagonal represents the accuracy and the underlined values are the highest accuracy per class. The others are the incorrect classification. The values are meant to be read in line.

3.3.3 Reliability

The objective of our method is to do segmentation applied for navigation. Besides, our classes are significantly related to each other since they describe, for three of them, a simplified depth information. As a result, there are edge effects that minimize the performances of the intersection over union (IoU) metrics. It is proposed to expand previous experiments with the evaluation of the methods reliability.

It is achieved by analyzing the confusion matrix presented in table 5. It permits to highlight the distribution of the true and the false positives per class, i.e. the areas misclassified from one class to another with regard to the ground truth. Values of the four methods are displayed in the same table. Thus, each cell contains four values and is ordered as indicated in the top left corner of the table. It is intended to be read in line, so each for each method, the distribution sum up to 100% per line. For example, the sparse method with ResNet 18 has its prediction of the *close* class being correctly matched (true positive) in 88.38% of the time, and wrongfully matched (false positive) as *middle* in 8.12% of the time, as *far* in 0.47% of the time and as *ground* in 3.04% of the time.

For navigation purposes, a high accuracy is required for the detection of the nearest obstacles. This task is primarily performed by the two classes *close* and *ground*. The *ground* class has 95% of true positives for each of the four methods, confirming the good IoU results. The *close* class has a score over 90% for the ResNet 50 methods and presents almost no critical cases. Indeed, two-thirds of the false positives indicate obstacles closer

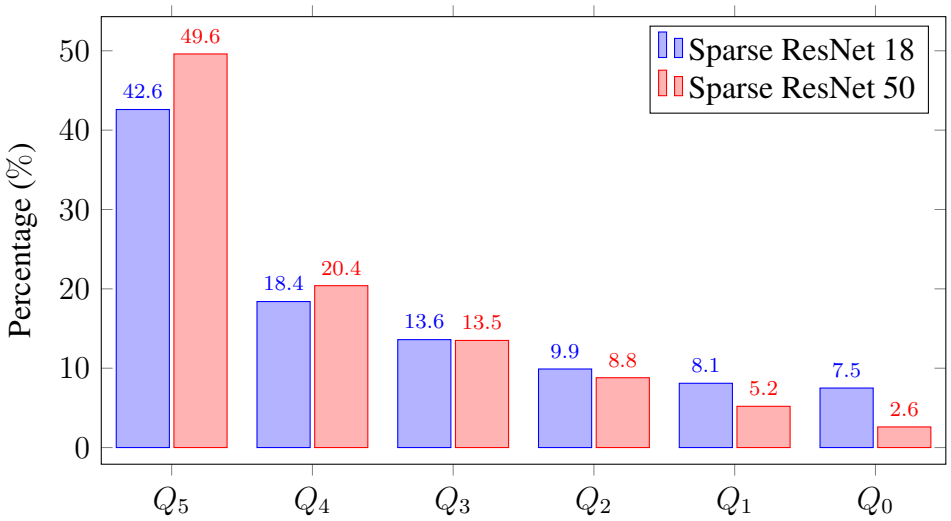

Figure 15. Quadtree distribution per level. Percentage of the images predicted by each layer of the decoder in average over a set of 700 images. Nearly 50% of a segmentation in the image can be predicted from the first layer of the decoder.

than they really are. The highest concern is for the *middle* class which shows over 20% of false positives. In practice, with such results, the *middle* class would have to be considered as an area to avoid, similarly to the *close* class. But it fulfills its initial goal, which is to be a buffer between the *close* and *far* obstacles and ensuring separation. Those two classes have a very low overlap with less than 3% of false positives between one another.

When comparing approaches to each other, the ResNet 50 based methods are the ones with the highest accuracy. Besides, for the two classes of interest *close* and *ground*, the best performances are obtained with the ResNet 50 sparse method, using QGN. It is, therefore, the most efficient of the four presented methods.

3.3.4 Network Complexity

The quadtree data structure allows compressing significantly the information. The Figure 15 presents the data distribution among each depth level of the quadtree, from Q_5 to Q_0. Between 40 to 50% of the information can be predicted by the first prediction layer of the decoder (Q_5). It means that the rest of the decoder only has to process less than 60% of the information. This percentage decreases after each prediction layer, leaving only a small portion of the data to be processed by the last layers. Since the decoder is composed on submanifold sparse convolutions [35], operations are only computed on active sites, i.e. information that have not been classified by previous prediction layers. Ideally, the two distributions must have been equal, as the predicted quadtree should be the same. But the differences in the predictions are inducing some biases, making the ResNet 50 proposing a more compressed representation.

Method	FLOPs	Parameters	Memory
Dense ResNet 18	24G	27.4M	0.78GB
Sparse ResNet 18	10G	26.5M	0.83GB
Dense ResNet 50	56G	72.3M	1.14GB
Sparse ResNet 50	19G	70.7M	1.27GB

Table 6. Networks complexity. Compare the FLOPs, the network parameters and the memory consumption of each method.

The subdivision into quadtree permits to drastically reduce the floating-points operations per seconds (FLOPs) compared to regular CNN approaches, as presented in table 6. Yet the memory usage remains equivalent due to the necessity to provide a dense prediction at full resolution, reconstructed from the quadtree layers. Note that the FLOPs are related to the quadtree size. Subsequently, it can fluctuate from one view to another. The number of parameters remains mathematically high and is almost equivalent to the dense counterpart due to the similarity in the architecture. Yet, for the same reasons as for the FLOPs, the prediction is highly sparse and all the parameters are unsolicited during the inference.

3.4 Summary

In this chapter, we have presented a segmentation framework dedicated to obstacle detection using a monocular navigation system. The proposed network architecture is derived from QGN and provides efficient prediction of a limited set of classes relevant for obstacle avoidance task. The reference segmentation map has been constructed from a stereo depth prediction network and converted into classes through geometric operations.

The experiments conducted on the Kitti dataset highlighted our method capability to achieve similar segmentation precision as the dense counterpart but with a reduced computational complexity. Indeed, the quadtree data structure permits to drastically reduce the density of the prediction. It was also demonstrated that most of the information can be extracted within the first few layers of the decoder justifying the usage of quadtrees that avoids unnecessary computations.

The method has been experimented with both ResNet 18 and ResNet 50 convolutional blocks. The quantitative results benefit the ResNet 50 solution, yet it is composed of almost three times more parameters. Therefore, it seems the sparse ResNet 18 is more appropriate for lighter applications. The trade-off between the network's complexity and the accuracy of the output would need to be discussed during the deployment of the solution on an autonomous driving system.

Chapter 4

Quadtree Depth for Navigation

This chapter explores the use of quadtree generation to obtain a compressed representation of a depth map to be used for navigation. The first part defines the framework of this navigation map. The second part presents its implementation in a self-supervised way. The third part presents the results obtained with this method.

4.1 The Navigation Map

This first section presents the objectives of the method and the impact of the quadtree compression applied to depth values.

4.1.1 Objectives

The precedent chapter presented a method to predict a segmentation map for obstacle avoidance. The limited number of classes permitted to fully benefit from the quadtree generating networks to obtain a compressed representation. Yet, detecting obstacles can be insufficient, as it does not permit a mapping of the environment. Therefore, the autonomous system might benefit from being able to extract more information about the scene's geometry by acquiring a depth map. But in practice, the same degree of precision is not required everywhere on the image. Besides, slight details are simply not relevant for the navigation task. Consequently, it is proposed in this chapter to present a method to estimate a compressed depth map that focuses on important details and on close range obstacles. The information is organized as a quadtree and predicted from a single image, as illustrated in Figure 16.

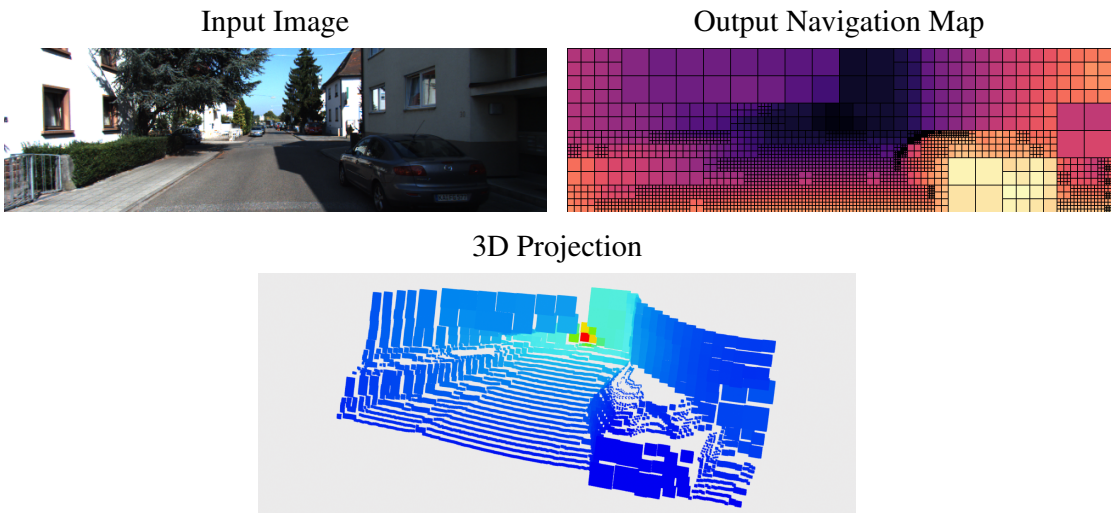

Figure 16. Monocular prediction overview of the N-QGN framework. At the top are the input image and the output depth map quadtree. At the bottom is the 3D projection of the quadtree depth information.

In many situations, it is required to manage compressed information as it permits to limit the size of the point cloud. Indeed, a dense depth map generates one information per pixel, which can be too expansive to process in real time for an embedded system. This solution is generally addressed by down sampling the information or by extracting close range depth values. While the first solution might induce to omit some significant information, the second is unsuitable for open areas such as urban scenes. The main observation to make is the same level of information is not required at close and at long range and must be adaptable. From this multi-scale free representation, it is possible to take advantage of the quadtree algorithm.

One could argue the quadtree can be constructed from the dense depth map to obtain the desired compressing and the appropriate data organization. But it would require to be post-processed, adding extra time to the computation pipeline. On the contrary, with the usage of QGN, the quadtree can be directly predicted saving computation time. Similarly than for the segmentation task, the depth prediction would benefit from the usage of sparse convolutions [34] to only process information required to construct the quadtree.

This implementation will combine the work developed on the quadtree segmentation task with self-supervised monocular depth methods. The similar QGN architecture as presented in Figure 11 in chapter 3 will be used, but trained using self-supervised monocular depth loss function. The novelty of the approach comes from the nodes subdivision decision. Contrary to the segmentation task, there are no classes, so no simple way to do the quadtree subdivision. The depth represents a continuous information, and a specific criterion must be defined to rule the manner the quadtree nodes will be subdivided.

4.1.2 Compressing the Monocular Depth Prediction

Depth information on the image will not serve the same purpose depending on its values. A good precision is required at short range as it permits to accurately detect upcoming obstacles to navigate safely. The long range depth helps to increase the map accuracy over time by providing rough estimation of far obstacles. Yet it is generally taken into account with a high uncertainty coefficient as the uncertainty grows with range as explained in section 3.1.2.3. Subsequently, the subdivision criterion will favor short-range accuracy and be based on the disparity value. The disparity is equivalent to the inverse of the depth, so a small disparity variation will have more impact at long range than at short range.

It is intended for this work to be completely self-supervised and only driven by the predicted depth values. Therefore, the quadtree must be constructed without the complete knowledge of the dense information. As a result, it will be constructed from the root of the tree, based on the predicted disparity information at that resolution. The disparity map will be scanned per block of four pixels to decide if more information have to be extracted at that location in the image. The nodes will be split if the highest difference between the four children values ($v_i, i = 1..4$) exceeds a threshold τ. On the image, it will consist of applying patches $\mathcal{A}$ of size 2x2 with a stride of 2, as illustrated in the following equation:

$$\mathcal{A} = \begin{cases} \mathbf{1}_{2\times2} & \text{if } \max(v_i) - \min(v_i) > \tau, \\ \mathbf{0}_{2\times2} & \text{else.} \end{cases} \tag{4.1}$$

The Figure 17 illustrates the behavior of the criterion on disparity and on depth values. The comparison is performed based on the pair of values $\min$ and $\max$ inside an area. If this pair is located under the curve, the nodes will be subdivided. A higher threshold value will result in a more restrictive criterion. The decision will define the activation map of the following layer in the decoder, but will have no impact on the already predicted values regardless of the criterion results.

4.2 Self-Supervised Training

This section presents the details of the unsupervised training phase resulting in the prediction of the navigation map. It will present the architecture of the network, the multi-scale training, the optimization of the monocular depth and the adjustments to be made for the specificity of a sparse prediction.

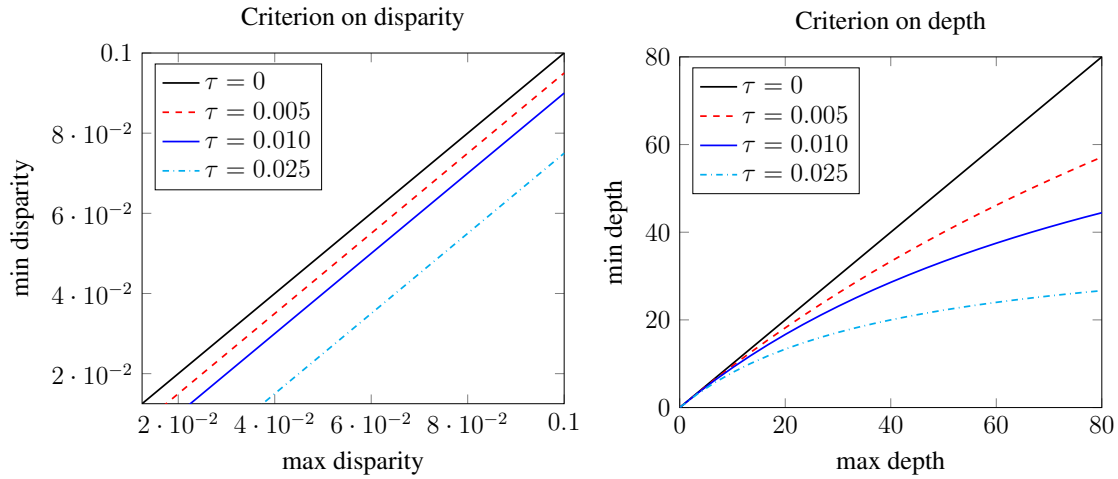

Figure 17. Quadtree subdivision criterion. The nodes are subdivided if the pair of values (max, min) intersect under the curve.

4.2.1 Network Architecture

The network is using the same architecture as the segmentation approach in chapter 3. It is a U-Net composed of a dense ResNet encoder and a sparse decoder based on QGN. From a single image, the method predicts a quadtree representation of the scene, corresponding to a multi-scale sparse depth prediction. The subdivision decision is deduced from the predicted depth information to decide the locations where more details have to be extracted in the image. This subdivision decision is represented by the activation maps A_l, computed based on the disparity values from Q_l as presented in Figure 18, with $l \in [1, 5]$ corresponding to the predicted quadtree level. At the beginning of each decoder layer, the activation map A_{l+1} calculated in the previous prediction step is used to define the set of active sites for the sparse feature map.

As presented in Equation 4.1, the subdivision criterion is set by threshold value which is defined prior training. This means that the network is trained and optimized to predict depth data at the compression level set by the threshold value. A new training is necessary to obtain data with another compression ratio. The criterion is such that a decrease in the value increases the compression rate, thus reducing the computational complexity.

4.2.2 Multi-Scale Prediction

The multi-scale depth map prediction method is particularly well fitted for quadtree prediction because it is in essence a multiscale representation. During training, the multi-scale photometric error minimisation loss from self-supervised depth prediction can be directly used by extracting multi-scale representation from a quadtree.

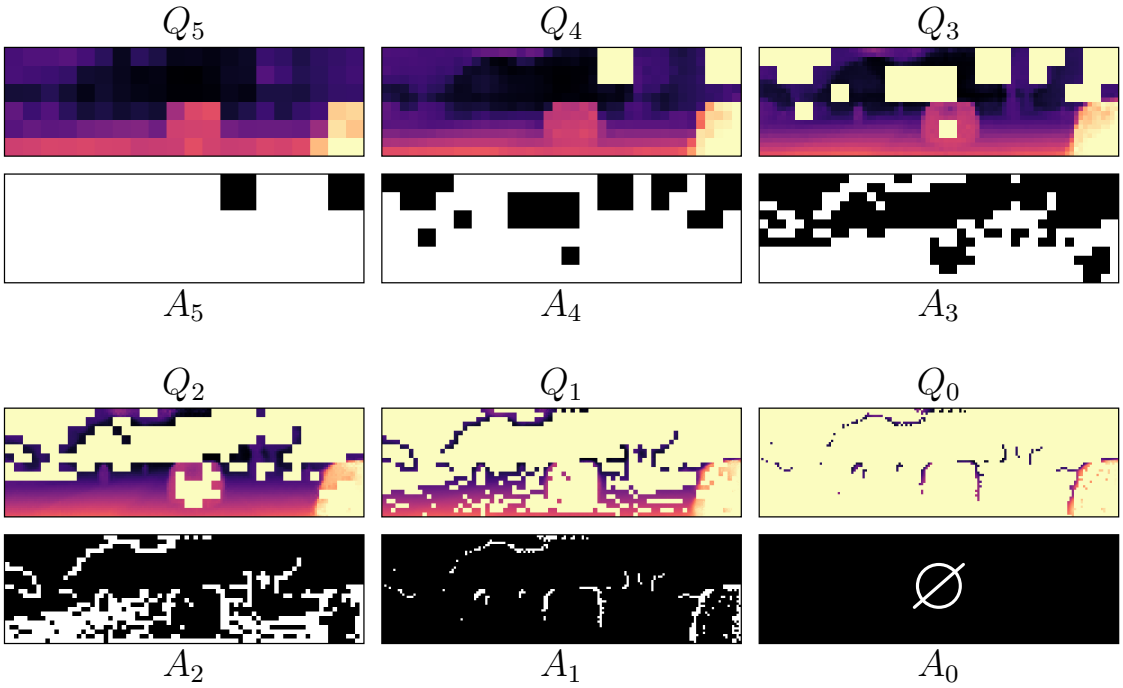

Figure 18. Output quadtree prediction decomposition. The network decoder is predicting the sparse depth maps Q_5 to Q_0. From each of them is deduced the corresponding activation map A_5 to A_1 for the next layer of the decoder. A_0 is not computed as it is predicted by the last layer.

As illustrated in Figure 18, the network is generating a multi-scale sparse prediction, with only the important details being computed. But the quadtree information can be combined to create a dense depth representation at the desired resolution. It consists of reconstructing the image by preserving the highest available resolution. This operation is defined in Equation 4.2 to combine data from the lowest resolution Q_5 to the highest desired resolution Q_n with $n \in \{4, ..., 0\}$.

$$Q_{5 \to n} = Q_n + \sum_{i=n+1}^{5} Q_i \cdot (1 - A_{i-1}) \tag{4.2}$$

where A_{i-1} is the active site map of Q_{i-1} with values in $A_{i-1} \in \{0, 1\}$. Values are kept from a specific level of the quadtree Q_i only on the inactive site locations, which correspond to the highest available resolution.

4.2.3 Monocular Training

This quadtree recombination permits to consider the predictions as a full pixel-wise multi-scale depth map. In that sense, we can apply the same loss function from [33], which is a

combination of a photometric error loss $\mathcal{L}_p$ and a per-pixel smoothness loss $\mathcal{L}_s$:

$$\mathcal{L} = \frac{1}{N}\sum_{i}^{N}(\mu\mathcal{L}_p^i + \lambda\mathcal{L}_s^i) \tag{4.3}$$

with N being the number of levels in the quadtree and μ and λ are constants value chosen empirically. In this approach, $N = 6$ as it is the sum of the loss function optimized on the prediction of Q_0 to Q_5. The values have been set to $\mu = 1$ and $\lambda = 10^{-3}$ as it is done in the litterature [32].

The photometric reprojection error, noted pe, between a source I_t and a target image $I_{t'}$ can be expressed as follows:

$$\mathcal{L}_p = \min_{t'} pe(I_t, I_{t'\to t}) \tag{4.4}$$

with $I_{t'\to t}$ being the projection of $I_{t'}$ into the frame t based on the knowledge of the predicted depth, the relative pose and the camera intrinsic parameters. The photometric error is a combination of L1 norm and SSIM such as

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1-\alpha)\|I_a - I_b\| \tag{4.5}$$

where α is a constant value set to 0.85 [32].

The smoothness loss in Equation 4.6 is sharpening the edges and is encouraging smooth values on textureless areas. As a result, the depth prediction will have smaller variations along the value. It leads to a higher compression of the quadtree as less subdivision is required to describe the scene.

$$\mathcal{L}_s = |\partial_x d_t^*|e^{-|\partial_x I_t|} + |\partial_y d_t^*|e^{-|\partial_y I_t|}, \tag{4.6}$$

with $d_t^* = d_t/\bar{d}_t$ the mean-normalized inverse depth to prevent the depth to diminish [82].

4.2.4 Encoder Pretraining

Correctly optimizing a quadtree prediction can be difficult. As explained earlier, it is a sparse prediction, meaning there is a part of the information that has not been computed. Thus, it is not possible to minimize the error on this missing part. Therefore, the optimization is limited to the actual predicted information. The issue has been addressed in the precedent chapter by guiding the supervision with data from the reference map.

But, in the case of a self-supervised training, there exist no reference map to use as guidance. Therefore, it is required to have a good first estimate to ensure that the optimization process will not diverge. It leads to the solution of pretraining the encoder to predict monocular dense depth maps. Therefore, it has already learn how to extract relevant depth features. Indeed, the latent space between a dense depth network and the N-QGN is highly similar as the extracted information is largely overlapping.

4.3 Experiments

In this section is discussed the capability of the end-to-end network to efficiently construct a quadtree navigation map and how it competes with equivalent quadtree constructed from dense information. Experiments were conducted on the Kitti dataset [31] as it is widely used for depth map prediction, permitting to be easily compared with the state of the art.

4.3.1 The Quadtree Navigation map

This method was the first to propose to directly generate a depth map as a compressed quadtree. We since developed a second approach which will be further discussed in the following chapter. For the time being, the performances of N-QGN will be compared to dense monocular depth prediction methods which were optimized with the same loss function. To evaluate the methods on a fair level of comparison, the dense depth map will be converted into quadtree with an equivalent level of compression. Indeed, the generated navigation map does not aim at out-performing dense prediction, but rather to propose an interesting trade-off between accuracy and compression.

With the usage of the quadtree subdivision decision function, the compression of the prediction can be adjusted before training. In this study, the experiments are conducted at

two levels of compression defined according to:

$$\text{Compression rate} = \frac{\sharp(D)}{\sharp(Q)} \tag{4.7}$$

where D and Q are the number of elements in the quadtree and in the dense prediction respectively.

Qualitative results of each evaluated methods are presented in Figure 19. To better appreciate the quadtree structure, it is overlayed on the prediction map. As a result, the part of the image reaching the highest resolution will appear black as the overlay will hide the data. Visually, the N-QGN seems to predict results highly similar to the ones generated from dense data and manage to provide a convincing depth prediction. The quadtree structure is consistent with the objectives. It gives less importance to the details, as desired, which allows a stronger compression in some areas without real loss of information. Yet, there is some limitations, as illustrated in the 5^{th} image, where the methods seem to fail to uniformly subdivide the wall on the right of the image.

4.3.2 Quadtree structure likelihood

The data structure in the quadtree is evaluated by taking as reference the structure obtained with LEAStereo. It is taken as a baseline a dense monocular approach converted into a quadtree.The structure likelihood is computed as presented in Equation 4.8 and consists of comparing the subdivision decision per node as described in the binary activation map A.

$$\mathbf{L} = \frac{1}{6}\sum_{i=0}^{5}\left(\frac{1}{N_i}\sum_{j=0}^{N_i-1}\left(1 - |a_{ij} - a^*_{ij}|\right)\right) \tag{4.8}$$

with a_{ij} and a^*_{ij} the j^{th} element of the binary activation maps of respectively the prediction A_i and the reference A^*_i which defines the quadtree structure. This function is defined such that a quadtree constructed randomly would score a correspondence of 50% with the reference. The results are presented in table 7 and highlight the capability of the method to generate quadtree structure. It obtains a convincing level of similarity regarding to the one constructed from dense information.

The likelihood score is providing a global indication of two quadtrees similarities. But it is compelling to analyse the data distribution among each level of the quadtree. The Figure 20 is composed of two graphs presenting the data distribution along the different levels of

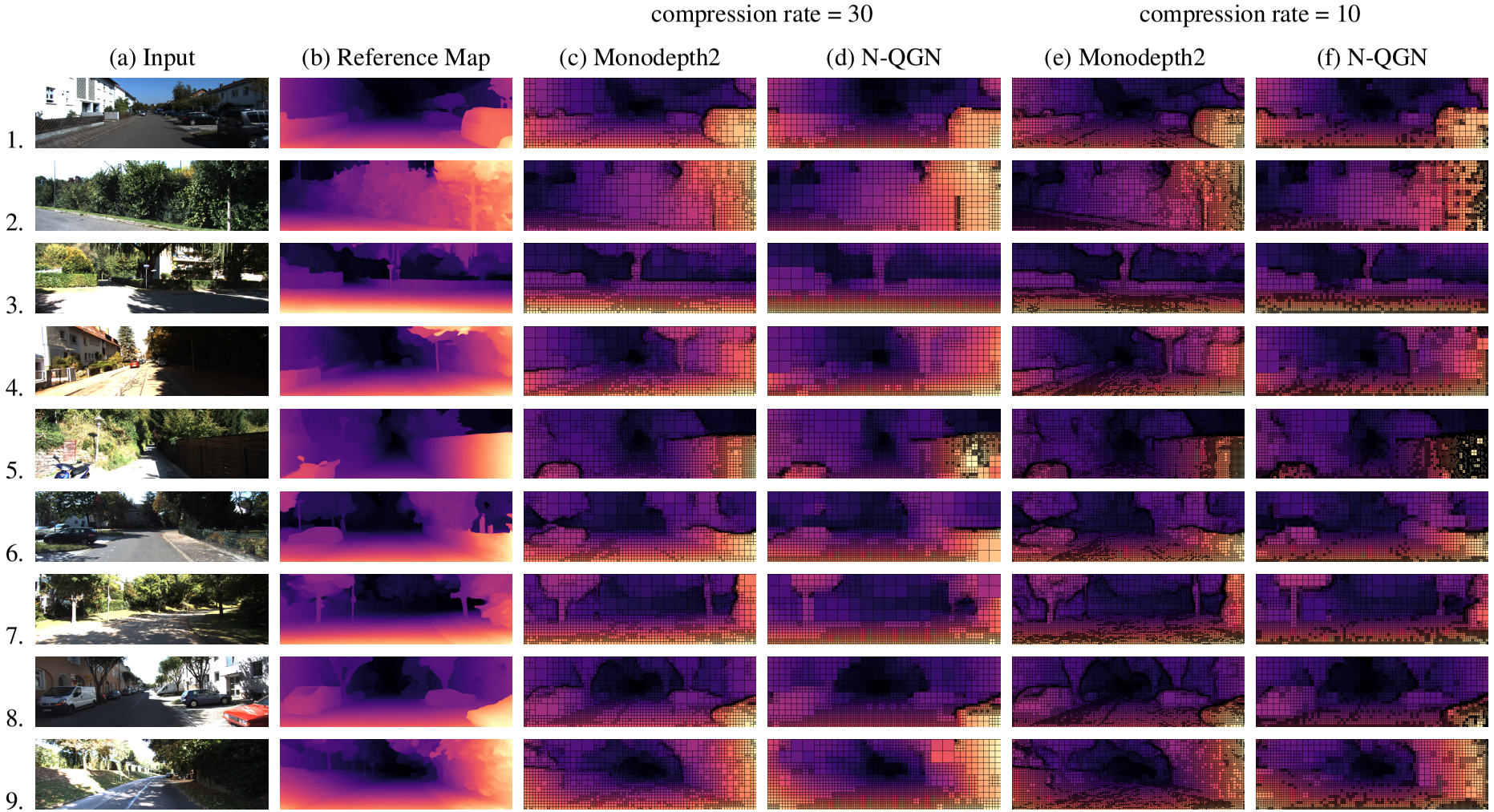

Figure 19. Qualitative prediction overview. From left to right, the single input image (a), the stereo reference map [13] (b), Monodepth2 [33] converted into quadtree compressed 30 times (c), N-QGN compressed 30 times (d), Monodepth2 [33] converted into quadtree compressed 10 times (e) and N-QGN compressed 10 times (f). The compression rate is averaged over a set of validation images and can vary from one image to another as it is dependent of the geometry of the scene.

Methods	compression rate	quadtree structure likelihood
monodepth2[33] + quadtree	30	**88.0%**
N-QGN (ours)	30	83.5%
monodepth2[33] + quadtree	10	**85.7%**
N-QGN (ours)	10	83.0%

Table 7. Evaluation of the quadtree prediction likelihood at two compression rate on the Kitti 2012 benchmark. The stereo network LEAStereo [13] is used as reference. Monodepth2 converted into quadtree is used as baseline.

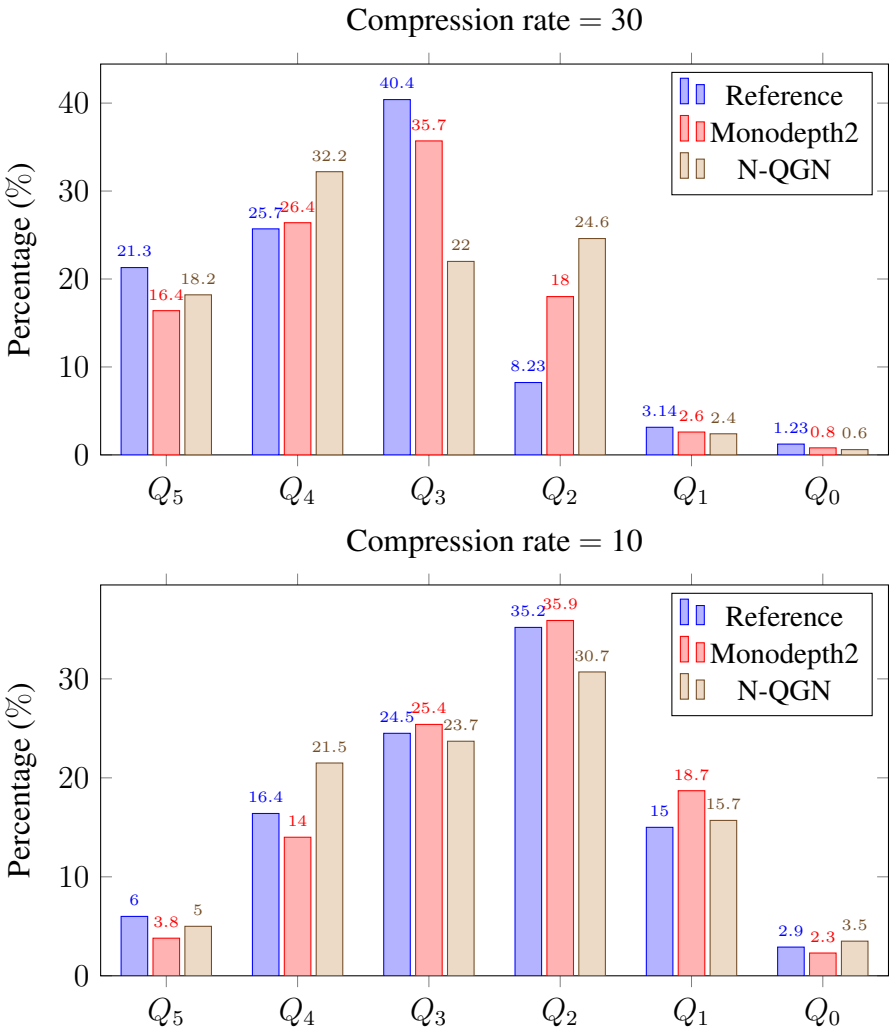

Figure 20. Quadtree distribution per level. Percentage of the images predicted by each layer of the decoder in average over a set of 700 images. On the top is the graph with the quadtree compressed 30 times, and on the bottom it is compressed 10 times. On each graph, the distribution of the reference LEAStereo, monodepth2 into quadtree and N-QGN (ours) are compared.

Methods	Comp. ratio	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
monodepth2 + quad.	30	0.125	**0.856**	5.309	0.204	0.843	0.950	**0.981**
N-QGN (ours)	30	**0.120**	0.928	**5.084**	**0.202**	**0.858**	**0.951**	0.978
monodepth2 + quad.	10	0.118	**0.802**	5.031	**0.196**	0.860	**0.955**	**0.981**
N-QGN (ours)	10	**0.116**	0.881	**4.946**	0.197	**0.867**	**0.955**	0.979
monodepth2	1	0.107	0.804	4.667	0.187	0.885	0.961	0.981

Table 8. Depth evaluation on the Eigen split [25] benchmark on the dataset Kitti.

the quadtree at the two compression rates. Each graph is comparing the presented method N-QGN with the reference map, used for the evaluation, and the monocular depth network [33] converted into quadtree, used as baseline. The percentage associated with each Q_i is the average percentage of information from that Q_i that is featured in the recombined quadtree. A significant percentage for one of the Q_i means that it largely describes the final quadtree.

For the quadtree with a compression rate of 30, only a thin portion of the image is predicted in the last two levels Q_1 and Q_0. It permits for a high reduction of the computational cost, compare to the dense methods. Surprisingly, the data distribution of N-QGN is different than the other two methods, permitting to have more data at Q_2 for an equivalent compression rate. This observation point that some areas have been over-subdivided due to the way the quadtree is constructed. For the compression rate of 10, the three methods are following the same distribution, with Q_5 and Q_0 representing a small portion of the image. The data is less compressed which can be seen visually with the majority of the data being predicted in Q_2 instead of Q_4/Q_3 with higher compression.

4.3.3 Depth Evaluation

The accuracy of predicted depth is evaluated on the Kitti dataset using the Eigen split benchmark [25]. It consists of evaluating the error with respect to the ground truth on a set of testing images which were not used during training. The error is measured using several metrics as presented in table 8. As for the quadtree distribution, the N-QGN method is compared with monodepth2 converted into quadtree at two levels of compression. The results of the dense monodepth2 are also listed to allow to contextualize the influence of the compression on the predictions.

Except for the square relative metric, which presents a significant loss, the N-QGN is achieving a higher accuracy at equivalent level of compression compare to monodepth2. Indeed, the quadtree network has been trained to predict depths at these compression

Methods	FLOPs	Parameters
monodepth2 [33]	8.0G	14.842M
N-QGN (comp. rate = 30)	5.3G	13.115M
N-QGN (comp. rate = 10)	5.7G	13.115M

Table 9. Comparison of model size and complexity

levels and proposes the proper value minimizing the error. In contrast, the compression in quadtree of the dense baseline is done by averaging values of the pixels at the compressed locations - which may cause an error increase. It is also notable that the loss in precision with respect to the dense method is reasonably small. This opens the discussion on the consequences of this decrease in real environment.

4.3.4 Memory Footprint Analysis

As explained before, the quadtree generating network is using sparse convolutions [34] to reduce the computation cost, by only computing operations on the active sites, i.e. the non-zero features storing data. Therefore, the complexity of the network is linked to the sparsity of the prediction. The Figure 20 illustrates the data distribution in the quadtree, between Q_0 to Q_5, and the resulting compression rate. Therefore, the more the information is compressed, the less floating point operations (FLOPs) have to be performed by the network. The table 9 presents the gain concerning the FLOPs induced by the compression. Sparse convolutions are only used in the decoder, making it the only one benefiting from this compression. It justifies the reason the reduction is not higher. Overall, the gain between the compression of 10 and 30 is not significant. Consequently, it seems more appropriate to work with the lower compression as it provides a more accurate depth prediction.

4.4 Summary

In this chapter, it was demonstrated the capability to directly predict depth under a quadtree data structure without a full knowledge of the dense information. The experiments have highlighted that the method is able infer an accurate compressed depth information, thus to provide a depth map for navigation. It takes advantage of the submanifold sparse convolution to eliminate superfluous information inside the network. Therefore, it is no longer necessary to use dense prediction to infer a quadtree. Sparse convolutions were initially designed to efficiently process incomplete or discrete data. In that sense, the data can be willingly sparsified to produce sparse representation from dense data to focus the interest on relevant information.

The monocular quadtree depth prediction is a double challenge. It combines the ill-posed problem of predicting the depth from a single image and the quadtree subdivision decision from a limited knowledge of the information. Yet, we have come up with a simple and efficient solution to address this challenge. Deciding for the subdivision upon the knowledge of the four direct children of each node allows to reduce the problem complexity. As a result, we have a self-supervised method presenting compressed, yet accurate performances.

Our study focuses on producing a light representation by deliberately choosing a strict subdivision criterion. An alternative would be to predict the most accurate quadtree depth map to compete with the state of the art. Indeed, the last layers of the decoder only deal with a fraction of the image, which are mostly edges. So, it makes them more specialized to efficiently predict a specific information.

Chapter 5

The Optimum Quadtree of a Depth Map

It is proposed in this chapter to extend the concept studied previously by suggesting a quadtree depth prediction compressed into an optimum quadtree structure as if it was constructed with dense knowledge. This allows to set new objectives, to discuss the modifications brought to the training procedure and to study the consequences on the prediction quality.

5.1 The Optimum Quadtree

This first section presents the new objectives to achieve with this framework as well as the changes applied to the subdivision criterion.

5.1.1 Objectives

It was demonstrated in the previous chapter the capability of a network to predict a quadtree representation of depth for navigation. The method focused on predicting a highly compressed representation by defining a very strict subdivision criterion. The quadtree was constructed inside the decoder by applying a criterion on the predicted disparity values. Therefore, no knowledge of the optimum quadtree was integrated in the optimization phase. Subsequently, there was no guarantee the training would converge to the optimal solution. In this chapter, it is proposed to integrate the prediction of the optimum quadtree subdivision into the network. This addition permits to achieve a different objective compared to the previous implementation. The knowledge of the optimum quadtree subdivision is bringing the depth prediction closer to the actual uncompressed dense depth prediction. Therefore, the study will investigate the difference of accuracy between the quadtree prediction versus

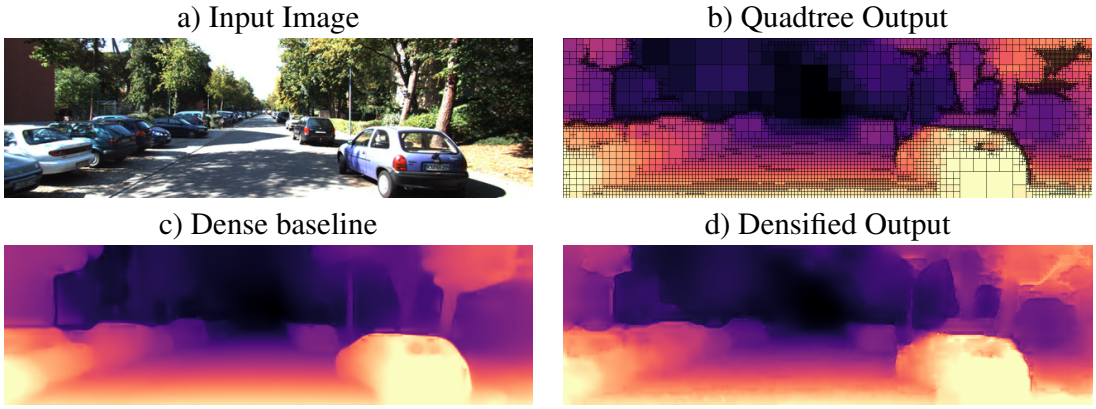

Figure 21. Method overview. *(a)* is the input RGB image, *(b)* is the predicted depth quadtree with a compression rate of 10, *(c)* is the dense prediction using monodepth2 [33] and *(d)* the densified quadtree depth from *(b)* with bilinear interpolation.

the dense depth prediction, as illustrated in Figure 21.

In this manner, it will be introduced in this network a probabilistic subdivision prediction to decide how to efficiently compress the quadtree. This probabilistic prediction will be supervised during training by a dense depth prediction network. For this purpose, the stereo depth network introduced in Chapter 3, regarding the segmentation approach, will be used as a training reference. It offers the advantage of constructing the reference quadtree structure with the complete knowledge of the pixel-wise depth values. The subdivision criterion can thus follow more advanced rules than the one used in N-QGN [5].

It is also proposed in this chapter to study the inference time of quadtree based networks. The gain in complexity and computational cost is undeniable, as shown previously. However, the use of sparse convolutions in this case may lead to slowdowns which can hinder a real-time execution. It is therefore proposed to explain these limitations and propose solutions to overcome them.

5.1.2 Probabilistic Subdivision Decision

The probabilistic subdivision decision is a map, predicted by the network along side the depth, to determine the set of active sites for the subsequent layer of the decoder. It is, in some sort, serving an equivalent purpose as the *mixed* class introduced in chapter 3 for the quadtree segmentation. This probabilistic map stores information for each pixel in the interval $[0, 1]$. The 0s representing the certainty of not subdividing and the 1s the certainty of subdividing the pixels. As this decision only allows two states, the predicted value will be binarized with the threshold value of 0.5. This probabilistic map will frequently be referred as the activation map noted A, as it defines the set of active sites for the sparse

convolutions in the decoder.

Contrary to the segmentation, the depth represents a continuous information, and gathering nodes per equal information is not likely going to compress the data. Consequently, this subdivision criterion has to be a trade-off between compression rate and depth accuracy. Similarly to the N-QGN approach in chapter 4, the criterion is applied on disparity values as they are more discriminative at short range. Therefore, even with a linear criterion, it will favor details at close range over unrelevant details at the back of the scene. Contrary to the N-QGN framework, the subdivision decision is to be learnt during the training. Therefore, it has to be supervised by a depth prediction method, to have access to the optimum subdivision. Subsequently, the stereo network, presented in chapter 3, will be used as a training reference.

The access to a dense supervision permits to construct the subdivision criterion on the knowledge of the full information. Consequently, it is based on the standard deviation of points in the area, noted $\sigma(p)$ with p a set of N pixels as described in equation 5.1, with $\bar{p}$ the average value in p.

$$\sigma(p) = \sqrt{\frac{\sum_{i=1}^{N}(p_i - \bar{p})^2}{N}}. \tag{5.1}$$

If the standard deviation σ exceeds a pre-defined threshold value, it would mean the disparity values are different enough to justify for this node to be subdivided into quadtree.

Ultimately, it is considered that beyond a certain distance, the depth prediction is no longer reliable, as presented in the Figure 8 in the chapter 3. Consequently, a maximum threshold is applied to the criterion to not search for details beyond a certain distance. On a set of pixels of disparity values, it means we are not seeking details if all values are lower than a threshold. In conclusion, the subdivision criterion is defined as follow:

$$\mathcal{A}(p) = \begin{cases} 1, & \textit{if } \sigma(p) > \tau \text{ AND } \max(p) < \lambda, \\ 0, & \textit{else,} \end{cases} \tag{5.2}$$

with τ and λ are two threshold values set before training. τ permits to adjust the accepted deviation and λ the maximum depth. They both define the compression rate of the predicted quadtree.

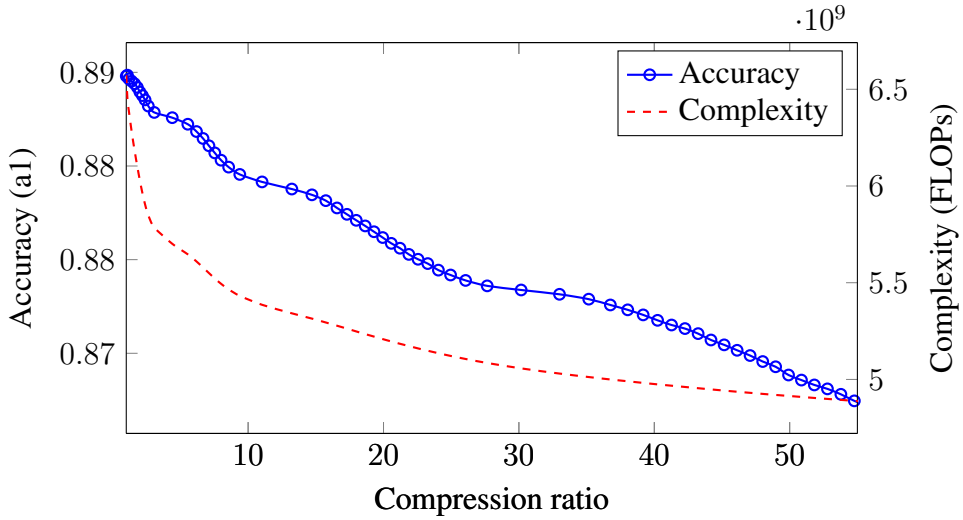

Figure 22. Theoretical evolution of the depth accuracy and network complexity with respect to the compression rate of the quadtree prediction. The compression criterion is applied to dense depth estimation framework to obtain an estimate of the expected accuracy. It also provides the data distribution in the quadtree from which results the complexity reduction.

5.1.3 Forcasting the Influence of the Quadtree Compression on the Depth Prediction

The experiments conducted on the previous methods have demonstrated the similarity of the results obtained between dense and sparse approaches. This allows to forcast the expected results of the sparse method by applying the compression criterion, presented in the section above, on a dense framework. The variation of the compression criterion permits to draw a trend curve presented in Figure 22. It also shows the gain induced on the complexity of the network. These data are deduced from the distribution of the data between the different levels of the quadtree, i.e. the spread over Q_5 to Q_0. As explained previously, once a piece of information has been predicted at a quadtree level, it is removed from the activation map. It is therefore no longer taken into account in the following layers of the decoder.

The monodepth2 framework [33], which presents similarities in its architecture with our sparse framework, is used as a baseline to plot the trend curve in Figure 22. The subdivision criteria, defined in Equation 5.2, is composed of the two variables λ and τ. They define the maximum depth and minimum disparity variations to look for more detail in the depth data, respectively. The curve in Figure 22 presents the evolution of $\tau \in [0, 0.006]$ with a progression step of 10^{-4}. The variable λ is set to a constant value of 50 meters, considering long range depth predictions are known to be unreliable as presented in Figure 8.

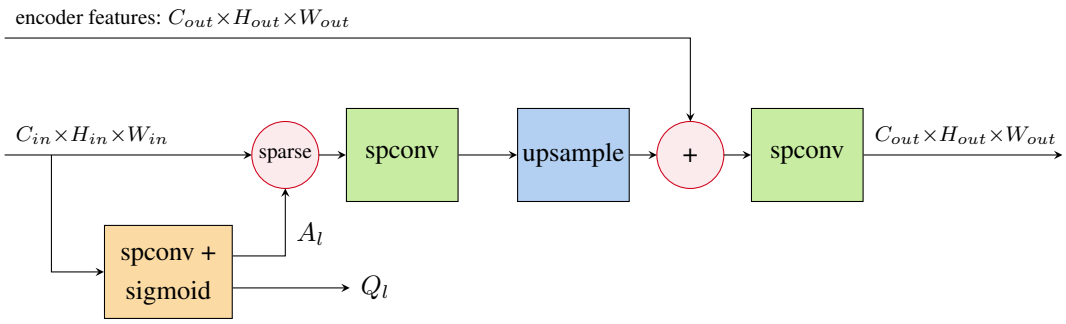

Figure 23. Decoder layer details.

The results unsurprisingly show a decrease in accuracy as the compression rate increases. Three inflection points are observed on the curve and are located around the values 4, 12 and 30. They show a slowing down of the loss of accuracy, corresponding to stability plates. At these locations, the gap between two measurements is greater, so the criterion λ adjustment gathers similar information in the image, having little impact on the accuracy. Outside these areas, the compression adjustment causes a real loss of information.

The evolution of the complexity of the network does not follow the same trend and proposes a drastic reduction in the number of FLOPs as soon as the compression is halved. Beyond this point, the slope of the curve slows down progressively and seems to approach an asymptotic value after the information has been compressed 50 times. Indeed, the optimization is only done on the decoder which represents 2.1 GFLOPs out of the 6.6 in total. In the end, to find the criterion that offers the best trade-off, we have to select the one with the largest gap between the two curves, which is the value 5.5 in this case. However, this is only an estimate of the theoretical behavior of the network. Therefore, the most interesting trade-off is probably located in a range of compression rate between 2 and 20. For the experiments, the two compression rates of 10 and 30 will be considered thereafter to evaluate the gain of the usage of a quadtree compression.

5.2 Supervised Training

The new subdivision criterion used to build the quadtree structure requires a reference map during training. This section details the changes in the network architecture, the optimization loss function, and the training guidance to improve the quality of the prediction.

5.2.1 Network Architecture

Similarly to precedent architecture presented in Figure 11, this N-QGNv2 architecture is based on an U-Net [64] composed of a dense encoder and a sparse decoder. To evalute the

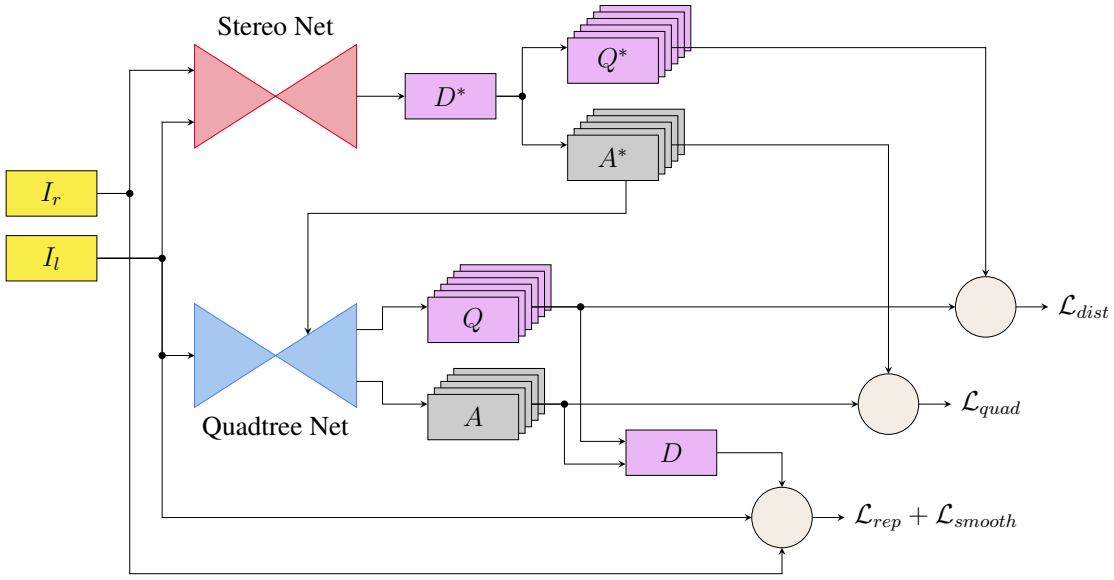

Figure 24. Training diagram of N-QGNv2. The Quadtree network learning is supervised by stereo network and is also taking advantage of self-supervised monocular training loss function. For more stability, the quadtree subdivision is guided by the reference map during training.

influence of the encoder onto the prediction, the experiments will be conducted on either with ResNet 18 [39], as used in N-QGN [5] or with MobileNetv2 [67], a lighter alternative. Concerning the decoder, two changes have been made. First, the activation map is now predicted by the network and used to sparify the feature map. Then, the short connection to feed the encoder features to the decoder is done between the two convolutions, just after the upsampling operations as illustrated in Figure 23. In N-QGN, it was original performed during sparsification operation in the following layer, to limit the switching operations between sparse and dense. Indeed, those operations are slowing down the execution as they require a data reindexing during the conversion from dense to sparse. However, it is possible to speed up the process by using on the data indexes given by the predicted sparse activation map. It is therefore possible to integrate the encoder information earlier, just after the upsampling operation, as it is normally done in a classical U-Net. The main advantage is that the features processed at a time when the information is less sparse. Therefore, more data can be taken into account by the network.

5.2.2 Loss Function

As illustrated in Figure 24, the loss function is composed of four terms as the network is predicting simultaneously two information: the depth information structured as a quadtree Q and the subdivision probability A. The problem has to be optimized on various levels.

At first, the optimization is made on the predicted depth values, by applying photometric

reprojection error $\mathcal{L}_{reproj}$ and smoothing $\mathcal{L}_{smooth}$ in self-supervised manner, respectively presented in equation 2.4 and 2.5 in section 2.2.2. They are evaluated on multi-scale with the densified disparity map D reconstructed from the set of Q and A.

In the mean time, the network is trained in a teacher/student relation, with a guidance composed of two terms. The teacher, or reference network, has been trained upstream to predict dense disparity information from stereo images as presented in section 3.1.3. The knowledge of dense depth information allows extracting the optimum quadtree representation fitting to the criterion defined in section 5.1.2. This dual depth and quadtree subdivision masks, noted respecively Q^* and A^*, are feeded to the network and used as ground truth during training. The disparity is optimized with the following distillation function $\mathcal{L}_{dist}$, initially presented in section 2.2.2:

$$\mathcal{L}_{dist} = \frac{1}{N_l} \sum_{i=1}^{N_l} \log(|d_i - d_i^*| + 1). \tag{5.3}$$

with d_i the disparity value of the i^{th} pixel in Q_l and d_i^* the reference disparity value from Q_l^* at the same location. The quadtree subdivision probability map A_l aims at minimizing the binary cross entropy function to fit the reference A_l^*:

$$\mathcal{L}_{quad} = \frac{-1}{N_l} \sum_{i=1}^{N_l} (a_i^* \log(a_i) + (1 - a_i^*) \log(1 - a_i)) \tag{5.4}$$

where a_i and a_i^* respectively corresponds to the predicted and ground truth probability value of subdivision of the i^{th} pixel in A_l and A_l^*. It can be noted a_i^* is storing binary values to represent the subdivision state of each pixel in the reference map.

Ultimately, the global loss function represents the weighted sum of the four previous terms averaged over the number of depth levels of the quadtree ($L = 6$ in our current approach).

$$\mathcal{L}_{global} = \frac{1}{L} \sum_{l=0}^{L-1} (\mathcal{L}_{dist} + \alpha \mathcal{L}_{quad} + \beta \mathcal{L}_{rep} + \gamma \mathcal{L}_{smooth}) \tag{5.5}$$

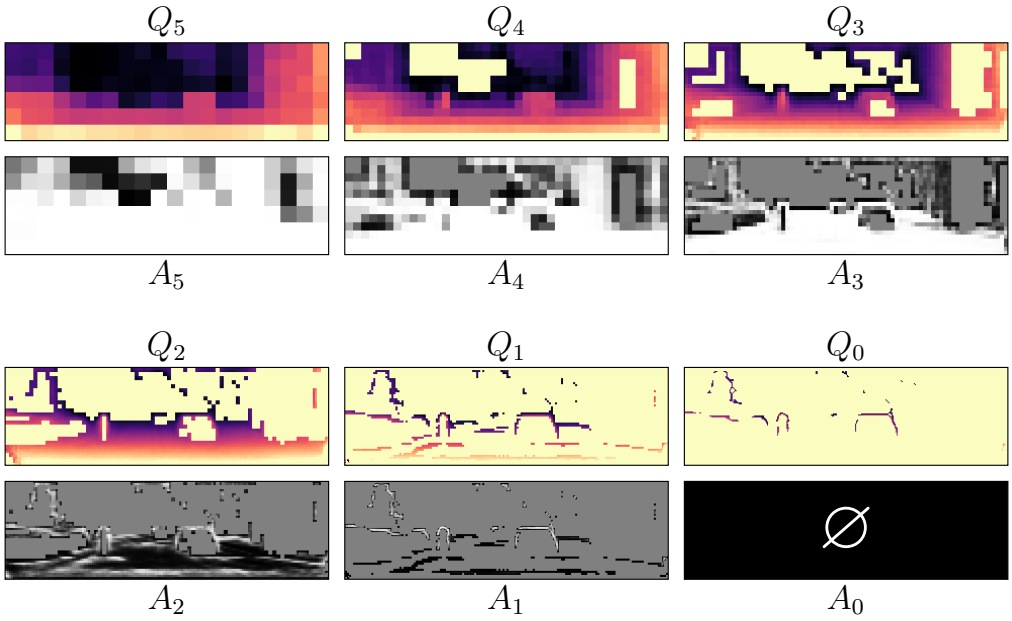

Figure 25. Output quadtree prediction decomposition. The network decoder is predicting the sparse depth maps Q_5 to Q_0, and the activation maps A_5 to A_1. A_0 is not computed as is predicted by the last layer. Pixels on the activation map with values over 0.5 are subdivided for the next layer of the decoder.

5.2.3 Guided Supervision

During the training phase, the subdivision is guided by the reference network similarly as it was done for the segmentation task in Chapter 3. As illustrated in Figure 24, the set of reference activation maps A^* is feeded to the QGN decoder, permitting to guide the network toward the optimum solution. At inference time, the guidance is removed, and the network is able to predict the quadtree representation without the help of any supervision. As explained before, it ensures the consistency and repeatability of the training.

Contrary to the segmentation approach, the learning of activation maps is here decorrelated from the rest of the data to be optimized. However, the depth optimization is dependent on the compression constraints imposed by the activation map. This means that the network is optimizing in parallel two semi-independent variables, whose convergence point can be different. Besides, as illustrated in Figure 25, the predicted information are more and more sparse as we advance in the decoder. But, it is not possible to take into account in the optimization process data that have not been predicted by the network. It implies that, without guidance, the training process may conclude the optimal solution consists in not subdividing the data beyond a certain layer in the decoder. This implies no predictions could be generated on subsequent layers and therefore no optimization is possible at these locations. This behavior is absolutely not desired and can be prevented with reference guidance.

5.3 Experiments

Experiments are conducted on the Kitti dataset [31] to evaluate the performances of this new N-QGNv2 framework. The objective is to measure the quality of the generated quadtree by considering the depth prediction and the tree structure. The inference speed will also be measured with comparison to dense equivalent methods and with N-QGN [5], presented in precedent chapter. As many parameters influence the quality of the prediction, the experiments are conducted at two levels of compression, with two different encoders and at two image resolutions. This multi-settings evaluation will permit to better consider the impact of the quadtree decoder on the prediction quality and speed.

5.3.1 Depth Estimation

The main objective of the network is to generate a compact representation of depth information using a quadtree structure. However, this compression comes with a cost, and the predictions may not be as accurate as those made with dense inference. Despite this, the loss is expected to be minimal to be a worthwhile trade-off for the increased speed.

The entire study is presented in tables 10 and 11, which correspond to methods using respectively a ResNet 18 and MobileNetv2 encoders. The first one is classically used for monocular depth prediction [33, 60] and the second one is designed for embedded network systems [15]. They are both presenting accuracy results by following standard evaluation metrics on the Eigen split Benchmark [23]. The methods are evaluated at resolution 192×640 and 320×1024. For the method with a ResNet 18 encoder, it is compared with EPCDepth [60], Monodepth2 [33] and our previous method N-QGN [5]. For the MobileNet, it is compared to a network trained by following standard monocular depth training.

The new framework is outperforming N-QGN by improving the accuracy and largely decreasing the runtime. As it will be further discussed in following sections, the gain in speed is due to the new decoder architecture and its implementation with an alternative library. Being able to predict the quadtree structure permits to avoid missing information as it could happen with N-QGN, implicitly reducing the error.

Even if a drop of accuracy can be observed compare to dense methods, the results are close and generally the second best when compared to the compressed versions. Interestingly, our method with a resnet 18 encoder, compressed 30 times, with an image size 320×1024 equals the performances of EPCDepth with image 192×640 while still being faster. This

Methods	Comp. ratio	H x W	fps↑	AbsRel↓	SqRel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
EPCDepth [60]	1	192x640	17.6	**0.099**	**0.754**	**4.490**	**0.183**	**0.888**	**0.963**	**0.982**
Monodepth2 [33]	1	192x640	**41.5**	0.108	0.820	4.693	0.188	0.884	0.961	0.981
N-QGN [5]	10	192x640	5.8	0.116	0.881	4.946	0.197	0.867	0.955	0.979
EPCDepth [60]	10*	192x640	17.6*	0.116	**0.731**	**4.624**	0.189	0.873	**0.962**	**0.983**
monodepth2 [33]	10*	192x640	41.5*	**0.107**	0.756	4.729	**0.188**	**0.879**	0.960	0.982
Ours	10	192x640	**41.9**	0.110	0.764	4.723	**0.188**	0.874	0.960	0.982
N-QGN [5]	30	192x640	6.0	0.120	0.928	5.084	0.202	0.858	0.951	0.978
EPCDepth [60]	30*	192x640	17.6*	0.119	**0.749**	4.784	0.192	0.864	**0.960**	**0.983**
monodepth2 [33]	30*	192x640	41.5*	**0.109**	0.763	4.846	0.190	**0.873**	0.958	0.982
Ours	30	192x640	**44.9**	0.113	0.792	**4.783**	**0.189**	0.871	**0.960**	**0.983**
EPCDepth [60]	1	320x1024	7.7	**0.093**	**0.671**	**4.297**	**0.178**	**0.899**	**0.965**	**0.983**
Monodepth2 [33]	1	320x1024	**17.9**	0.104	0.775	4.562	0.191	0.887	0.959	0.981
EPCDepth [60]	10*	320x1024	7.7*	**0.102**	**0.657**	**4.294**	**0.178**	**0.898**	**0.966**	**0.984**
monodepth2 [33]	10*	320x1024	17.9*	0.107	0.741	4.524	0.186	0.886	0.962	0.982
Ours	10	320x1024	**21.0**	0.104	0.702	4.532	0.183	0.887	0.962	0.983
EPCDepth [60]	30*	320x1024	7.7*	**0.103**	**0.656**	**4.346**	**0.178**	**0.895**	**0.966**	**0.984**
monodepth2 [33]	30*	320x1024	17.9*	0.107	0.737	4.569	0.186	0.883	0.962	0.982
Ours	30	320x1024	**25.1**	0.104	0.706	4.523	0.181	0.887	0.963	0.983

*Dense methods are geometrically converted into quadtree by applying the same criterion used to train ours. The conversion time is not considered in the frame per second (fps) results.

Table 10. Depth accuracy metrics of methods trained with a ResNet 18 encoder. Results on Kitti dataset [31] using the Eigen split evaluation [24] and are organized by compression rate. **Bold** values are the best score of the category and underlined are the second best.

Methods	Comp. ratio	H x W	fps↑	AbsRel↓	SqRel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
MobileNetv2	1	192x640	**46.8**	0.133	1.044	5.293	0.213	0.836	0.946	0.976
MobileNetv2	10*	192x640	46.8*	**0.132**	**0.967**	**5.274**	0.213	**0.831**	**0.945**	0.977
Ours	10	192x640	**49.8**	0.133	1.004	5.319	**0.210**	**0.831**	**0.945**	**0.978**
MobileNetv2	30*	192x640	46.8*	**0.134**	**0.965**	5.375	0.215	0.825	0.943	0.976
Ours	30	192x640	**50.6**	0.135	1.003	**5.315**	**0.211**	**0.829**	**0.944**	**0.978**
MobileNetv2	1	320x1024	**19.3**	0.122	0.987	5.051	0.200	0.859	0.954	0.980
MobileNetv2	10*	320x1024	19.3*	**0.120**	0.896	**4.940**	0.198	**0.859**	**0.955**	**0.981**
Ours	10	320x1024	**26.2**	**0.120**	**0.881**	5.001	**0.195**	0.858	**0.955**	**0.981**
MobileNetv2	30*	320x1024	19.3*	**0.120**	**0.886**	**4.973**	**0.198**	**0.856**	**0.955**	**0.981**
Ours	30	320x1024	**29.7**	0.125	0.937	5.073	0.201	0.850	0.952	0.980

*Dense methods are geometrically converted into quadtree by applying the same criterion used to train ours. The conversion time is not considered in the frame per second (fps) results.

Table 11. Depth accuracy metrics of methods trained with a mobilenetv2 encoder. Results on Kitti dataset [31] using the Eigen split evaluation [24] and are organized by compression rate.

Methods	Encoder	192 x 640		320 x 1024	
		comp. 10	comp. 30	comp. 10	comp. 30
EPCDepth	ResNet 18	84.4 %	88.2 %	84.4 %	87.5 %
monodepth2	ResNet 18	87.8 %	90.3 %	**86.2 %**	88.6 %
N-QGNv2	ResNet 18	**88.4 %**	**90.9 %**	**86.2 %**	**89.8 %**
MobileNetv2	MobileNetv2	87.3 %	89.8 %	85.7 %	88.5 %
N-QGNv2	MobileNetv2	85.6 %	89.5 %	83.7 %	87.5 %

Table 12. Evaluation of the quadtree structure likelihood at two compression rate and at two image size on the Kitti dataset [31]. The likelihood is computed with respect to the stereo network reference used for training. The dense methods are converted into quadtree and used as baseline. The highest performances are in **bold**.

is where quadtree compression is interesting. It struggles to be competitive on equal terms because of the constraints applied. But it is able to work with larger images at low cost which allows to compensate the loss due to compression.

The same applies to the results with the mobilenetv2 encoder. The gain in execution time remains limited for inference on low resolution images, but becomes interesting at high resolution by going from 19.6 to 29.7 frames per second. Surprisingly, the data compression also permits to improve results on the RMSE and RMSE Log metrics. It can be observed on the monodepth2 results at high resolution in Table 10 and on the MobileNetv2 results in Table 11. This can be explained by the fact that compression can bring a smoothing on the data, which can lead to a decrease of the prediction errors.

Ultimately, it is important to note that the computation times for the dense methods compressed in quadtree are given as an indication. It does not take into account the time needed for post-processing for compression. This widens the gap in execution time for anyone wishing to work with quadtree information.

5.3.2 Quadtree Subdivision

5.3.2.1 Data Distribution

The quadtree compression implies the depth information is distributed into various levels of the hierarchical tree. This data distribution is presented in Figure 26 and is compared with N-QGN [5] and the stereoNet reference used for training. For a same compression rate, the distribution is relatively similar, except for the N-QGN, as the subdivision criterion is different. Nonetheless, they all have the last level Q_0 only representing small areas in the image, which typically corresponds to object edges.

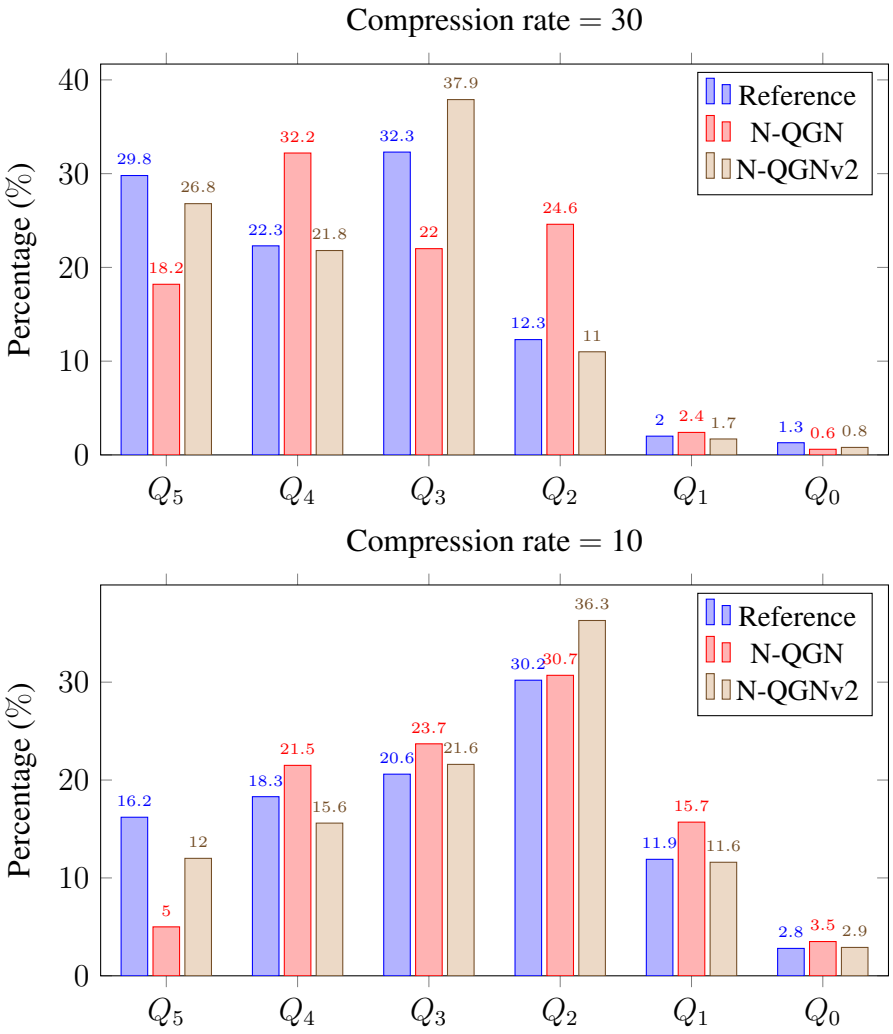

Figure 26. Quadtree distribution per level. Percentage of the images predicted by each layer of the decoder in average over a set of 700 images. On the top is the graph with the quadtree compressed 30 times, and on the bottom it is compressed 10 times. On each graph, the distribution of the reference StereoNet, N-QGN and our new framework N-QGNv2 are compared.

The data distribution provides some interesting information about the scene geometry. The subdivision criterion based on disparity value deviation applies a threshold to a maximum depth. It results in the information in the quadtree being separated by following this criterion. The first levels Q_5 to Q_3 are mainly flat areas with slight local deviations. The last levels Q_2 to Q_0 represent edges in the images, which are principally areas with the highest uncertainty. In addition, at high compression rate, these last layers are mostly ignored to merely represent a few percentages of the information in the depth map.

5.3.2.2 Subdivision Prediction

For this approach, the subdivision decision is predicted by the network along side the depth. It has been learnt based on the subdivision computed on the reference stereo network by applying the defined criterion. Its precision has been evaluated and is oscillating between 85 and 90% of likelihood to the reference quadtree, computed with knowledge of the dense information. It is computed based on the following equation:

$$L = \frac{1}{5}\sum_{i=1}^{5}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}\left(1 - |\bar{a}_{ij} - a^*_{ij}|\right)\right) \tag{5.6}$$

with $\bar{a}_{ij}$ and a^*_{ij} the j^{th} element of the binary activation maps of respectively the prediction A_i and the reference A^*_i, which defines the quadtree structure. The function is such that it considers equally each prediction A_i to balance the influence of the highest resolutions containing more pixel values. A quadtree constructed randomly would score a correspondence of 50% with this metric.

These results highlight the ability of the network to predict a coherent quadtree partitioning. The prediction of the structure instead of a deduction, as it was done in the precedent work permits to improve the results. One can notice that, in Table 12, the method with the highest compression rate provides a slightly better likelihood. This comes from the fact the information is more compressed, implying the last layers' predictions are mostly composed of zeros values, as less information are predicted there. The construction of a quadtree is a recursive process whose results depend at each level on the results of the previous one. Therefore, the results building the last levels of the quadtree, corresponding to the last layers of the decoder, are necessarily more uncertain and subject to more error.

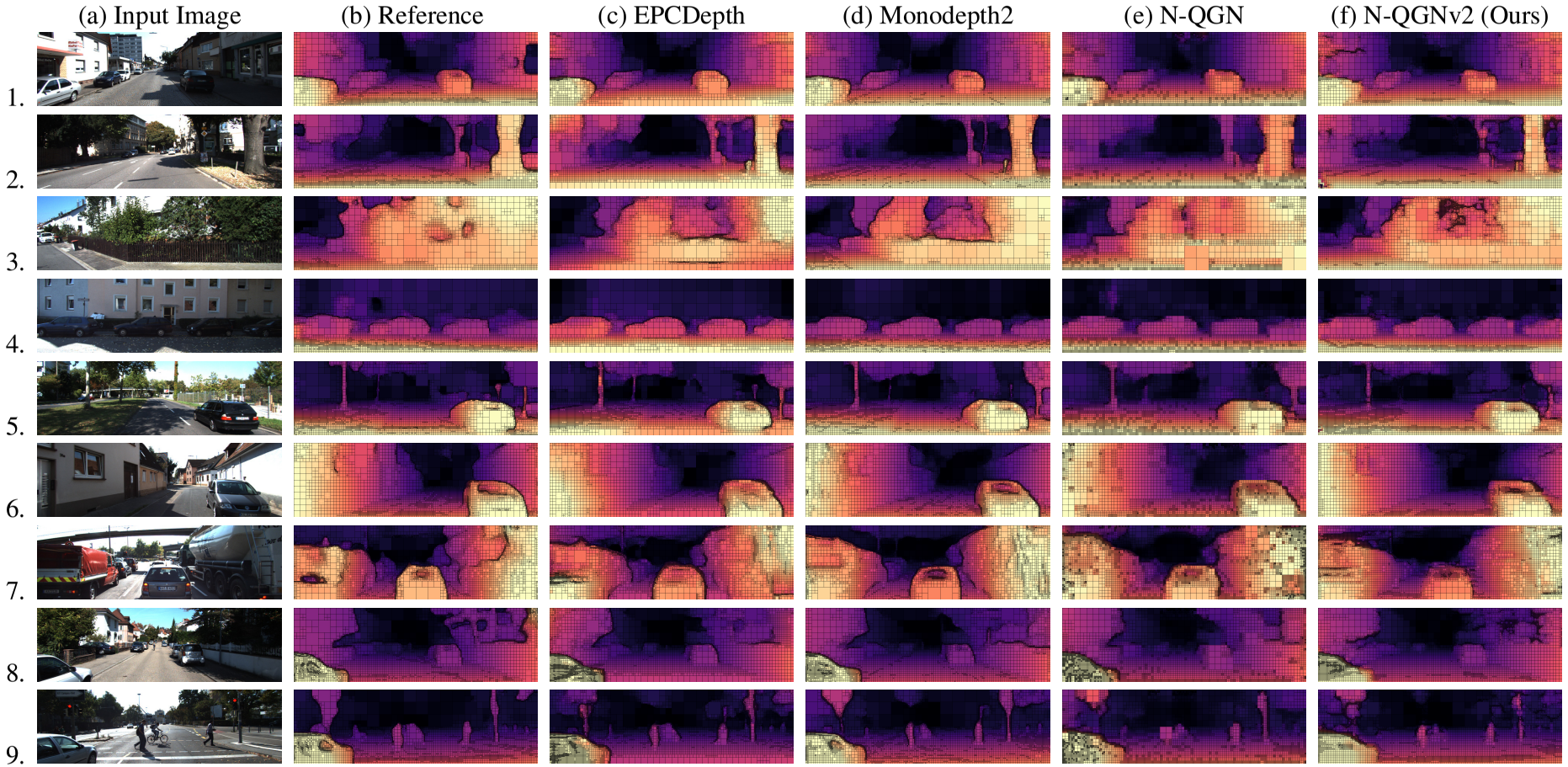

Figure 27. Qualitative quadtree depth prediction overview. From left to right (a) the input image, (b) the stereo reference map used for training, (c) EPCDepth converted into quadtree, (c) Monodepth2 converted into quadtree, (d) the N-QGN method and (e) our N-QGNv2 method. Results are displayed with a compression rate of 10.

5.3.3 Qualitative Results

Qualitative results are presented in Figure 27, comparing the new framework with the reference map, EPCDepth [60], the monodepth2 [33] and N-QGN [5]. To increase the readability of the quadtree structure, the border of each node is represented by black lines.

The nine images tend at proposing various scenario to evaluate the behavior of the method. Due to the geometry of the outdoor scenes and the presence of a vanishing point in the images, some areas like the road cannot be highly compressed. Some other parts, like close obstacles or fronto-parallel surfaces are highly compressed as they offer small disparity deviations in the values. This can be observed more particularly on images 3 to 5 with a high compressions on the cars and on the hedge facing the road. Besides, a strong compression is also observable in the background of the images, where the depth value is high, as the subdivision criterion is comparing disparity variations. Thus, the compression in the image largely depends on the geometry of the scene, because it is related to the distance and orientation of the objects.

Concerning the predicted quadtree structure, our approach proposes a coherent representation with respect to the one from the reference map. Some inconsistencies can still be noted on bushes in the image 3, as it appears to be the case for most methods. Compared to previous version N-QGN, the prediction is smoother and less subject to over-subdividing nodes, as can be observed on the road areas or more specifically on images 7 and 9 from Figure 27. Ultimately, it seems challenging to tell apart the depth maps obtained directly through our adaptation of the Quadtree Generating Network and the one constructed from dense information.

5.3.4 Runtime Evaluation and Software Limitation

Over the entire studies conducted in this thesis, it has been made clear the quadtree generating network permits to reduce the computational cost. The data can be compressed to limit the amount of operations to compute. Most importantly, this compression is not degrading too much the prediction quality.

The runtime experiments have been performed on a computer equipped with a GPU Nvidia Quadro P620 and a process Intel Core i7-9850H CPU at 2.60GHz $\times$ 12. This configuration might not represent the reality of embedded systems, but provides a base of comparison between dense and quadtree based inference methods. Results are presented in Table 10 and 11, next to the corresponding accuracy metrics. The runtime is dependent of the data

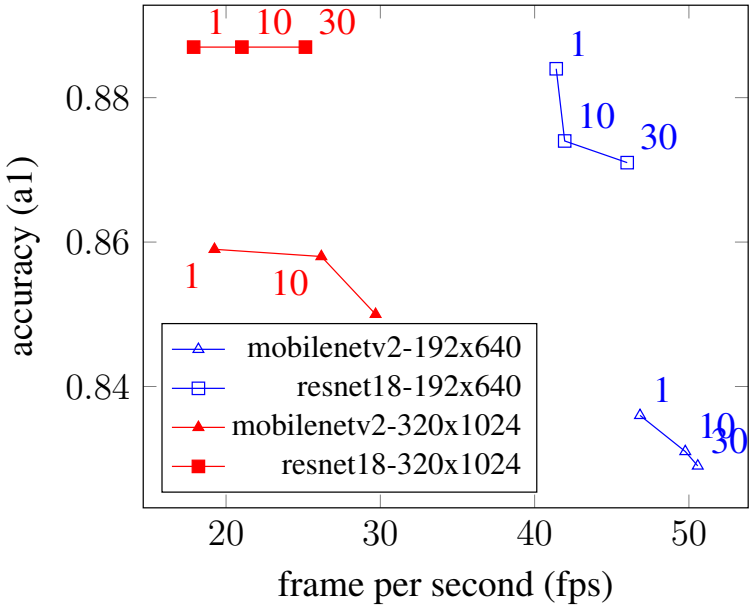

Figure 28. Runtime with respect to the encoder, compression rate and input image resolution. The label values on the point represent the compression rate.

compression and the choice of encoder.

Comparison with N-QGN [5] is not featured in the figure ahead, as it was demonstrated in table 10 the new framework is working faster. It is due to the improvements made on both the architecture and the implementation. This new framework is predicting how the nodes in the quadtree are subdivided, which goes faster than the previous solution consisting of geometrically computing it from partial data. Besides, the framework has been implemented with the *SpConv* [86] library providing a heavily-optimized sparse convolution solution and goes faster than *SparseConvNet* [34] previously used. In addition, we optimized the dense-to-sparse operation which was running slow due to the re-indexing procedure.

Figure 28 is highlighting the trade-off proposed by the usage of quadtree solutions. They are all proposing faster inference with lower accuracy, even if for the solutions with a resnet18 encoder, the gain in speed is minimal at low resolution. With Quadtree Generating Network (QGN), the gain in speed is only made possible by implementing a sparse decoder. Therefore, it has been decided to evaluate the impact a lighter encoder such as mobilenetv2 [67] could have on the runtime. It is, as expected, infering faster and is even able to reach 50 fps with our framework at low resolution, but comes at the cost of loss in accuracy.

5.4 Summary

This new method has demonstrated its efficiency to infer a direct quadtree representation of the scene. It allows focusing the interest on the most significant parts of the images to reduce the computing cost, with a minimal loss of accuracy. Compared to the previous method, the new framework is able to infer a faster and more accurate quadtree. It was made possible by predicting the proper way to construct the quadtree, instead of computing it based on partial depth data. The new implementation has also drastically reduced the inference time. The method is also proposing an interesting trade-off between speed and accuracy, especially for systems interested to work with depth represented as quadtree. The method is easily adaptable to any other architecture and consists of replacing the dense decoder with a sparse alternative.

The conducted experiments once again demonstrated the potential of quadtree generating networks. The usage of sparse convolutions permits to reduce the computation cost of the state of the art architectures without significant loss of accuracy. Even if the gain is slight at low resolution, it becomes genuinely interesting with bigger input images. The capability of the network to be able to efficiently predict how the quadtree nodes should be subdivided opens up prospects for more advanced subdivision criteria. One could consider taking into account the semantic of the scene in the subdivision criterion to have a depth resolution depending on which class it belongs.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

It was proposed in those works to explore the capability of the Quadtree Generating Networks applied to navigation tasks. It offers the double interest of allowing the generation of compressed data organized as a quadtree and of reducing the network computational cost through the use of sparse convolutions. These benefits are particularly interesting for navigation approaches. The same degree of precision is not required everywhere in the image, and a fast application capable of operating in real time is desired.

The study is organized around three works enriching the understanding of the scene from a single image, as illustrated in Figure 29. Based on monocular depth estimation works, these methods respectively propose a scene segmentation for obstacle avoidance and two depth maps for navigation. The first depth prediction method seeks an aggressive compression

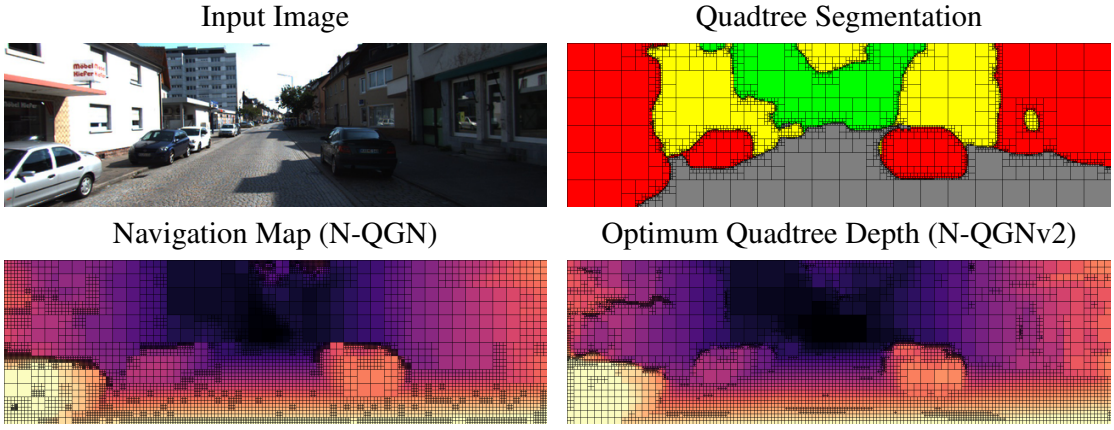

Figure 29. Methods prediction overview applied on the same input image.

of the information, at the risk of losing details. The second aims at learning the optimal quadtree structure according to advanced criteria.

The approach infering a segmentation map into a quadtree led to the exploration of two interesting notions: The generation of a quadtree using a sparse decoder and the construction of an unsupervised segmentation scheme. Experiments have shown the same level of accuracy can be achieved with this sparse method than with its dense equivalent. The difference is the segmentation map prediction can be greatly compressed, drastically reducing the computational requirements. Besides, the choice was made to propose the segmentation adapted to the problems of obstacle avoidance. It was generated from depth information and therefore did not require manual annotation of the images. This allows the segmentation from unsupervised information, making the method more flexible to other environments.

The second approach, called N-QGN, proposes to transpose the use of QGN from segmentation to depth prediction. It is recognized that to navigate, the same level of accuracy is not needed everywhere in the image and that very fine details are not important. Therefore, depth data can be compressed while remaining relevant for navigation. The network has been trained in a self-supervised manner by applying a constraint on the predicted data to iteratively build the quadtree. The experiments permitted to validate the feasibility and reliability of the method to achieve similar results as if constructed from dense equivalent framework, while being lighter.

The third approach, called N-QGNv2, aims at predicting the optimum quadtree depth. As for the precedent method, the information is still compressed, thus suitable for navigation application. However, the quadtree structure is inferred by the network instead of being deduced from partial depth information. It permits to learn advanced subdivision criterion, generated from dense information. The study has also been expanded to different image sizes and encoder architectures to evaluate their impact on the prediction accuracy and runtime. Even if the compression into quadtree induces a loss of accuracy, the gain in speed is noticeable, especially with high-resolution images.

Ultimately, these studies permitted to explore an alternative path to the dominant trend in deep neural networks. Dense architectures are not the only possible options as it may be beneficial to make them sparse to better address a given problem. This may lead to a loss in accuracy, but the gain in computational cost may be beneficial. It comes down to finding the perfect trade-off.

6.2 Perspectives

The introduction of sparsity in the deep neural networks permits to reduce the FLOPs, thus speeding the process at reduced cost, with a limited loss of accuracy. Yet, with the QGN framework, the encoder is based on dense convolutional architecture. With the sparsity of the decoder, a lot of features are pruned during the quadtree construction. Subsequently, numerous of features computed in the encoder ended up being unused. From this fact, one could imagine to rethink the entire architecture, in order to make it completely sparse for an extensive reduction of computation with a restricted loss of accuracy.

The sparsity in the networks architecture is becoming extensively popular as illustrated by the work of Kurtz et al. [47] on the subject. They are demonstrating the capability of sparse neural networks to compete with popular dense models such as ImageNet for the image classification, running only on CPUs. This corroborates the fact that not all connections in the network are relevant for the inference, and some of them can be pruned without suffering from significant loss. As of today, similarly to our approaches, the sparse networks are derived from dense networks to offer a trade-off between computation cost and accuracy. But it is fair to assume new architecture will emerge that are natively designed for sparse convolutions. As such, they permit either achieve different tasks or compete the dense approaches. In any case, with the emergence of giant architectures requiring massive amounts of GPU memory, it is obvious that not everyone will have the financial means to follow this path. Therefore, sparse architecture seems to be a good alternative to bring neural networks to a reasonable size.

6.3 Future Works

6.3.1 The Icosahedron Grid to Represent the Sphere

In future works, it may be interesting to expand this solution to other modalities, such as the omnidirectional images. Indeed, the gain in popularity of 360° vision cameras has drastically decrease their cost. Subsequently, they are now intensively studied in computer vision research. They are small enough to be carried on a drone and to provide a complete image of the scene, similar to a sphere. However, for practical reasons, this sphere is reprojected on a plane, called equirectangular projection. The coordinates of this plane correspond to the longitudes and latitudes of the sphere, which gives an image strongly distorted at the poles as illustrated in Figure 30. These strong distortions in the image impose the use of convolutions adapted to the sphere when using convolutional neural networks [18].

a) Sphere b) Equirectangular Projection

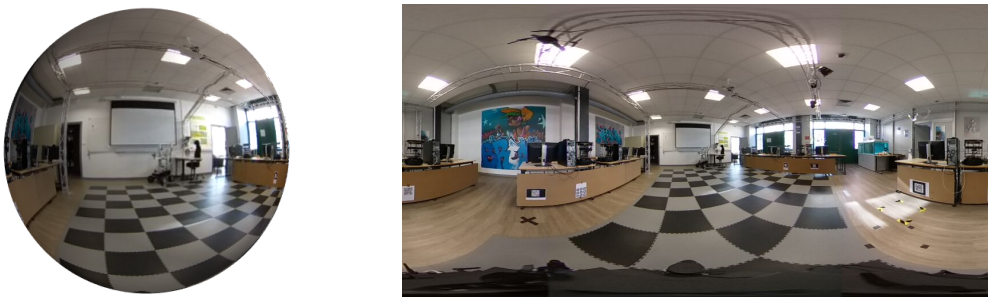

Figure 30. Spherical image projected into the equirectancular plane.

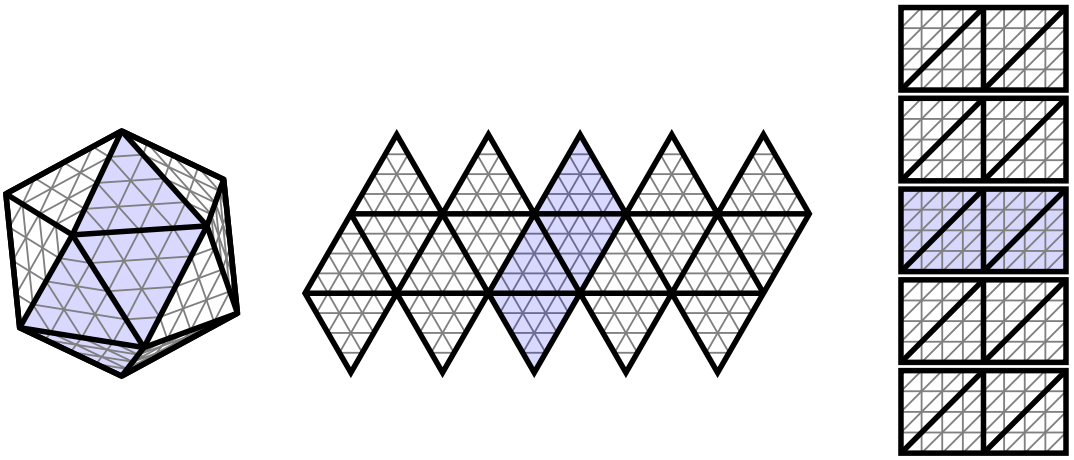

Figure 31. From the icosahedron to the regular 2D grid. From left to right, the 3d representation of the icosahedron, its unfolded representation and its transformation according to a regular mesh.

However, other modes of representation exist, such as the cubemap [78] or the icosahedron mapping [19]. The first one is approximating the sphere as a cube, projecting the scene into 6 squares. it has the advantage of correcting the distorsions in the image, making it compatible with most of the processing tools used on perspective images. However, it has the disadvantage of sampling the image unevenly, grabbing more details at the square corners than at the center. Another solution consists of projecting the sphere into the icosahedron, illustrated in Figure 31. It is a regular polyhedron composed of twenty triangular faces, on which are projected the spherical image. It has the advantage of correcting distortions while offering an almost perfect regular grid on the image. It correspond to a grid H_0, at its lowest resolution, with points located on the twelve vertices of the polyhedron itself. The resolution can be increased to the grid H_1 by splitting each triangle into four sub-triangles, introducing 3 new points located at the middle of each edges of the original triangles. Due to adjacency, some points are present on two triangle, bringing the number of points to 42 instead of 72. This process can be repeated n times to generate a grid H_n composed of $N = 5 \cdot 2^{2n+1} + 2$ points [19].

However, the hexagonal mesh of the icosahedron makes it difficult to manipulate the image. Nonetheless, it has been shown in [19] that it is possible to perform a transformation on the icosahedron to bring the hexagonal grid back to a regular grid with some minor adjustments. Indeed, the pixels present on the vertices of the polyhedron (thick black lines) have the specificity of being in the adjacency of 5 other pixels, while the others follow a hexagonal grid. This particularity will have to be taken into account when processing the image. The resulting regular grid is composed of five blocks of $2^n \times 2^{n+1}$ pixels, which with the padding will correspond to a shape of $5 \times (2^n + 2) \times (2^{n+1} + 2)$.

6.3.2 From the Quadtree to the Heptatree

The idea behind the use of the icosahedron to return to a regular grid is to develop a solution allowing to adapt the work explored with Quadtree Generating Networks on the sphere. The equirectangular projection, which is more frequently used for these approaches, does not fit to a quadtree data structure. On the contrary, the regular grid obtained with the icosahedron allows us to consider this solution. However, this grid is only regular for computational convenience, but it does represent an information following an hexagonal grid. Therefore, the choice is made to explore the possibility of working on a heptatree data strucutre, as illustrated in Figure 32.

The way the H_n grid is organized and how it subdivides from one resolution to another does not permit to use a standard heptatree. Indeed, the nodes are interconnected and would result in a structure organized as presented in Figure 32. It implies the nodes, except from the central ones, are linked to two fathers' nodes. The subdivision decision, would then have to depend on the values of these two nodes.

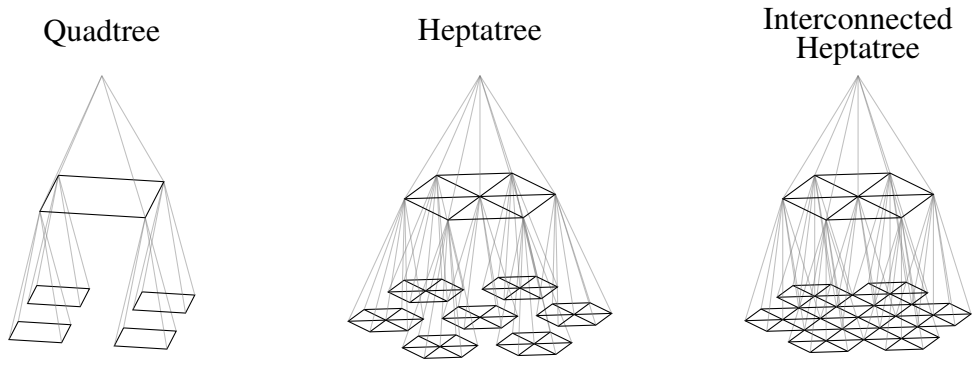

Figure 32. Quadtree, Heptatree and Interconnected Heptatree

Appendix A

Quadtree Decoder Details

The N-QGNv2 architecture, presented in Chapter 5, is composed of a dense ResNet 18 encoder and a sparse quadtree decoder detailed in Figure 13. The prediction layers are infering the Q and A, which correspond respectively to the disparity and activation maps at the given resolution. The predicted activation map is used to sparsify the feature map for the upcoming layer. As such, the decoder is composed of sparse convolutions manipulating sparse feature maps, With the exception of the one of the first prediction layer, which processed the dense features from the encoder.

Layer	Operations	Number of Filters	Kernel Size	Output Size
-	Input features	-	-	6×20×512
Prediction 5	Dense Conv	2	1×1	6×20×2
0	Sparse Conv	256	3×3	6×20×256
	Up Sample	1	1×1	12×40×256
	Sparse Conv	256	3×3	12×40×256
Prediction 4	Sparse Conv	2	1×1	12×40×2
1	Sparse Conv	128	3×3	12×40×128
	Up Sample	1	1×1	24×80×128
	Sparse Conv	128	3×3	24×80×128
Prediction 3	Sparse Conv	2	1×1	24×80×2
2	Sparse Conv	64	3×3	24×80×64
	Up Sample	1	1×1	48×160×64
	Sparse Conv	64	3×3	48×160×64
Prediction 2	Sparse Conv	2	1×1	48×160×2
3	Sparse Conv	32	3×3	48×160×32
	Up Sample	1	1×1	96×320×32
	Sparse Conv	32	3×3	96×320×32
Prediction 1	Sparse Conv	2	1×1	96×320×2
4	Sparse Conv	16	3×3	96×320×16
	Up Sample	1	1×1	192×640×16
	Sparse Conv	16	3×3	192×640×16
Prediction 0	Sparse Conv	1	1×1	192×640×1

Table 13. Quadtree decoder details for the N-QGNv2 with an input image of size 192×640.

Appendix B

Prediction Quadtre Decomposition

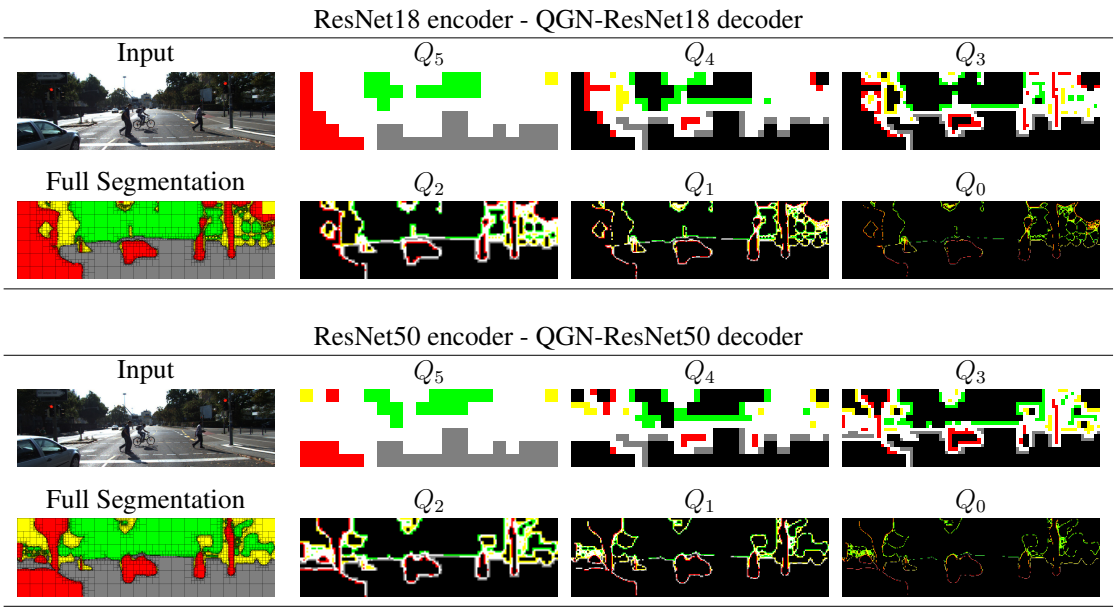

Figure 33. Qualitative results of the quadtree decomposition of the segmentation framework with ResNet18 and ResNet50 architectures.

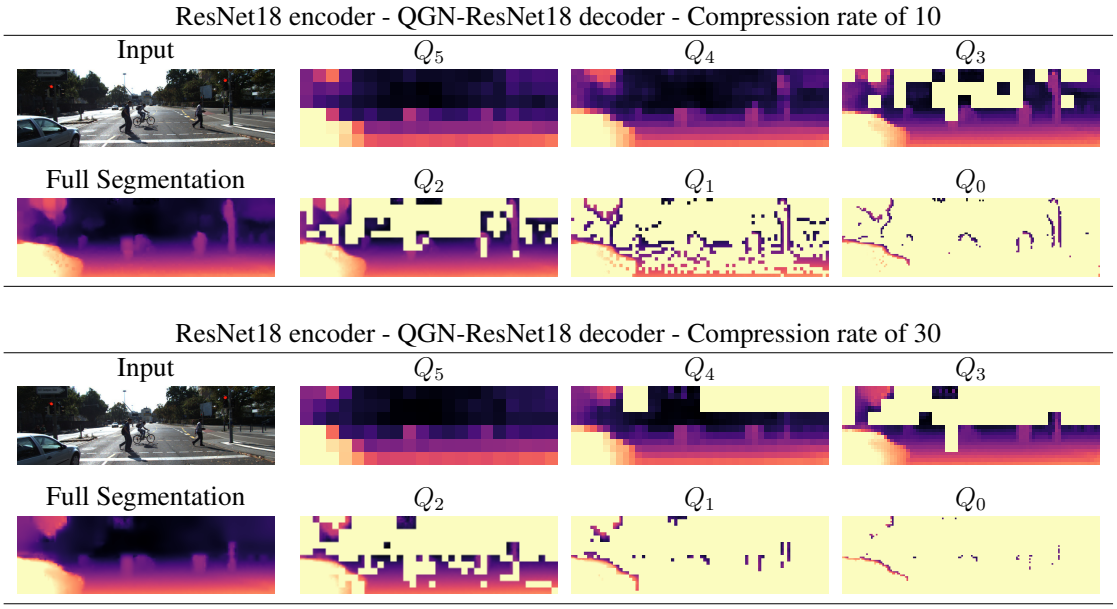

Figure 34. Qualitative results of the quadtree decomposition of the N-QGN framework with the ResNet18 architectures at compression rates 10 and 30.

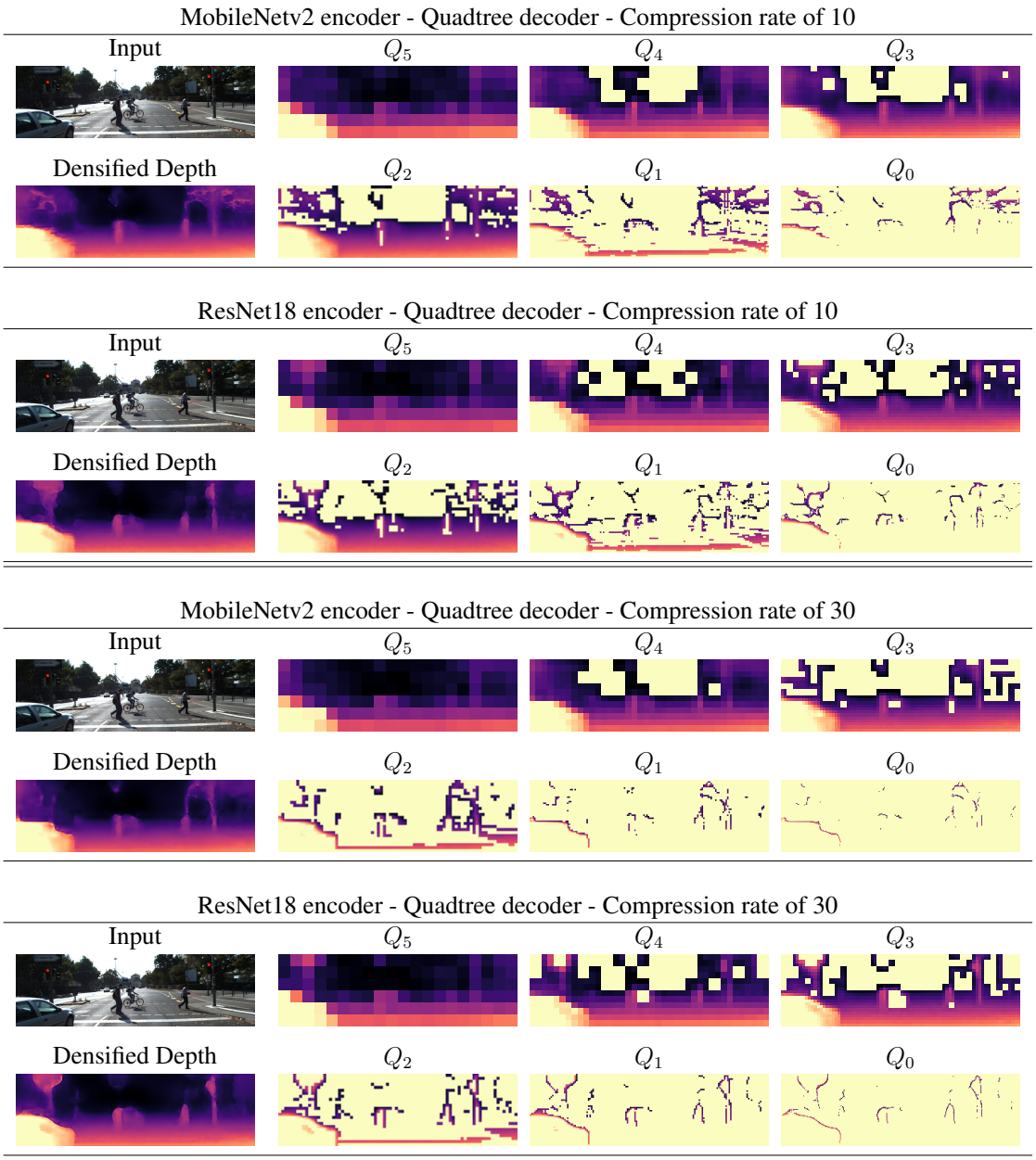

Figure 35. Qualitative results of the quadtree decomposition of the N-QGNv2 framework with the MobileNetv2 and ResNet18 architectures at compression rates 10 and 30.

References

[1] Md Zahangir Alom et al. “The history began from alexnet: A comprehensive survey on deep learning approaches”. In: *arXiv preprint arXiv:1803.01164* (2018).

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[3] Jongbeom Baek, Gyeongnyeon Kim, and Seungryong Kim. “Semi-Supervised Learning with Mutual Distillation for Monocular Depth Estimation”. In: *arXiv preprint arXiv:2203.09737* (2022).

[4] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. “AdaBins: Depth Estimation Using Adaptive Bins”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 4009–4018.

[5] Daniel Braun et al. “N-QGN: Navigation Map from a Monocular Camera using Quadtree Generating Networks”. In: *arXiv preprint arXiv:2202.11982* (2022).

[6] Jia-Ren Chang and Yong-Sheng Chen. “Pyramid stereo matching network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5410–5418.

[7] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[8] Liang-Chieh Chen et al. “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[9] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[10] Zhi Chen et al. “Revealing the Reciprocal Relations Between Self-Supervised Stereo and Monocular Depth Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15529–15538.

[11] Zhi Chen et al. “Revealing the Reciprocal Relations Between Self-Supervised Stereo and Monocular Depth Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 15529–15538.

[12] Bowen Cheng et al. “Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12475–12485.

[13] Xuelian Cheng et al. “Hierarchical Neural Architecture Search for Deep Stereo Matching”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[14] Kashyap Chitta, Jose M Alvarez, and Martial Hebert. “Quadtree generating networks: Efficient hierarchical scene parsing with sparse convolutions”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020.

[15] Yu-Chen Chiu et al. “Mobilenet-SSDv2: An improved object detection model for embedded systems”. In: *2020 International conference on system science and engineering (ICSSE)*. IEEE. 2020, pp. 1–5.

[16] Mian-Jhong Chiu et al. “Real-time Monocular Depth Estimation with Extremely Light-Weight Neural Network”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 7050–7057.

[17] Jaehoon Cho et al. “A large rgb-d dataset for semi-supervised monocular depth estimation”. In: *arXiv preprint arXiv:1904.10230* (2019).

[18] Taco S Cohen et al. “Spherical cnns”. In: *arXiv preprint arXiv:1801.10130* (2018).

[19] Taco Cohen et al. “Gauge equivariant convolutional networks and the icosahedral CNN”. In: *International conference on Machine learning*. PMLR. 2019, pp. 1321–1330.

[20] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[21] George R Cross and Anil K Jain. “Markov random field texture models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1983), pp. 25–39.

[22] Xingshuai Dong et al. “Towards Real-Time Monocular Depth Estimation for Robotics: A Survey”. In: *arXiv preprint arXiv:2111.08600* (2021).

[23] David Eigen and Rob Fergus. “Predicting Depth, Surface Normals and Semantic Labels With a Common Multi-Scale Convolutional Architecture”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[24] David Eigen and Rob Fergus. “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2650–2658.

[25] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth map prediction from a single image using a multi-scale deep network”. In: *arXiv preprint arXiv:1406.2283* (2014).

[26] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[27] Jun Fu et al. “Adaptive context network for scene parsing”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6748–6757.

[28] Jun Fu et al. “Dual attention network for scene segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3146–3154.

[29] Alberto Garcia-Garcia et al. “A survey on deep learning techniques for image and video semantic segmentation”. In: *Applied Soft Computing* 70 (2018), pp. 41–65.

[30] Ravi Garg et al. “Unsupervised cnn for single view depth estimation: Geometry to the rescue”. In: *European conference on computer vision*. Springer. 2016, pp. 740–756.

[31] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[32] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. “Unsupervised monocular depth estimation with left-right consistency”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279.

[33] Clément Godard et al. “Digging into self-supervised monocular depth estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3828–3838.

[34] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks”. In: *CVPR* (2018).

[35] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. “3d semantic segmentation with submanifold sparse convolutional networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9224–9232.

[36] Vitor Guizilini et al. “Semantically-guided representation learning for self-supervised monocular depth”. In: *arXiv preprint arXiv:2002.12319* (2020).

[37] John A Hartigan and Manchek A Wong. “Algorithm AS 136: A k-means clustering algorithm”. In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108.

[38] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[39] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[40] Armin Hornung et al. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[41] Ching-Pai Hsu et al. “A review and perspective on optical phased array for automotive LiDAR”. In: *IEEE Journal of Selected Topics in Quantum Electronics* 27.1 (2020), pp. 1–16.

[42] Zilong Huang et al. “Ccnet: Criss-cross attention for semantic segmentation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 603–612.

[43] Lam Huynh et al. “Lightweight Monocular Depth with a Novel Neural Architecture Search Method”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 3643–3653.

[44] Alex Kendall, Matthew Grimes, and Roberto Cipolla. “Posenet: A convolutional network for real-time 6-dof camera relocalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[45] Salman Khan et al. “Transformers in vision: A survey”. In: *ACM Computing Surveys (CSUR)* (2021).

[46] Ralf Kohler. “A segmentation system based on thresholding”. In: *Computer Graphics and Image Processing* 15.4 (1981), pp. 319–338.

[47] Mark Kurtz et al. “Inducing and Exploiting Activation Sparsity for Fast Inference on Deep Neural Networks”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. Virtual: PMLR, 13–18 Jul 2020, pp. 5533–5543. URL: http://proceedings.mlr.press/v119/kurtz20a.html.

[48] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. “Semi-supervised deep learning for monocular depth map prediction”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6647–6655.

[49] Jae-Han Lee and Chang-Su Kim. “Monocular depth estimation using relative depth maps”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9729–9738.

[50] Jun Liu et al. “MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 166 (2020), pp. 255–267.

[51] Wei Liu, Andrew Rabinovich, and Alexander C Berg. “Parsenet: Looking wider to see better”. In: *arXiv preprint arXiv:1506.04579* (2015).

[52] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[53] Donald Meagher. “Geometric modeling using octree encoding”. In: *Computer graphics and image processing* 19.2 (1982), pp. 129–147.

[54] S. Mahdi H. Miangoleh et al. “Boosting Monocular Depth Estimation Models to High-Resolution via Content-Adaptive Multi-Resolution Merging”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 9685–9694.

[55] Shervin Minaee et al. “Image segmentation using deep learning: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* (2021).

[56] Yue Ming et al. “Deep learning for monocular depth estimation: A review”. In: *Neurocomputing* 438 (2021), pp. 14–33.

[57] Pushmeet Kohli Nathan Silberman Derek Hoiem and Rob Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. In: *ECCV*. 2012.

[58] Angshuman Parashar et al. “SCNN: An accelerator for compressed-sparse convolutional neural networks”. In: *ACM SIGARCH computer architecture news* 45.2 (2017), pp. 27–40.

[59] Vaishakh Patil et al. “Don’t forget the past: Recurrent depth estimation from monocular video”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6813–6820.

[60] Rui Peng et al. “Excavating the Potential Capacity of Self-Supervised Monocular Depth Estimation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15560–15569.

[61] Matteo Poggi et al. “Towards real-time unsupervised monocular depth estimation on cpu”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 5848–5854.

[62] Xiaojuan Qi et al. “Geonet: Geometric neural network for joint depth and surface normal estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 283–291.

[63] Jiaxiong Qiu et al. “Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3313–3322.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[65] German Ros et al. “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3234–3243.

[66] Hanan Samet. “The quadtree and related hierarchical data structures”. In: *ACM Computing Surveys (CSUR)* 16.2 (1984), pp. 187–260.

[67] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[68] Daniel Scharstein and Richard Szeliski. “High-accuracy stereo depth maps using structured light”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 1. IEEE. 2003, pp. I–I.

[69] Ahmed Rida Sekkat et al. “SynWoodScape: Synthetic Surround-view Fisheye Camera Dataset for Autonomous Driving”. In: *arXiv preprint arXiv:2203.05056* (2022).

[70] Clifford A Shaffer and Hanan Samet. “Optimal quadtree construction algorithms”. In: *Computer Vision, Graphics, and Image Processing* 37.3 (1987), pp. 402–419.

[71] Suvash Sharma et al. “Semantic segmentation with transfer learning for off-road autonomous driving”. In: *Sensors* 19.11 (2019), p. 2577.

[72] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[73] Shiyu Song and Manmohan Chandraker. “Robust scale estimation in real-time monocular SFM for autonomous driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1566–1573.

[74] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), pp. 1–11.

[75] Haotian Tang et al. “Searching efficient 3d architectures with sparse point-voxel convolution”. In: *European conference on computer vision*. Springer. 2020, pp. 685–702.

[76] Bo Tao et al. “Self-supervised monocular depth estimation based on channel attention”. In: *Photonics*. Vol. 9. 6. MDPI. 2022, p. 434.

[77] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. “Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2088–2096.

[78] Fu-En Wang et al. “Bifuse: Monocular 360 depth estimation via bi-projection fusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 462–471.

[79] Kaixuan Wang, Wenchao Ding, and Shaojie Shen. “Quadtree-accelerated real-time monocular dense mapping”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[80] Linda Wang, Mahmoud Famouri, and Alexander Wong. “DepthNet Nano: A highly compact self-normalizing neural network for monocular depth estimation”. In: *arXiv preprint arXiv:2004.08008* (2020).

[81] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm. “Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5555–5564.

[82] Yang Wang et al. “Occlusion aware unsupervised learning of optical flow”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4884–4893.

[83] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[84] Jamie Watson et al. “The temporal opportunist: Self-supervised multi-frame monocular depth”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1164–1174.

[85] Mark Weber et al. “Step: Segmenting and tracking every pixel”. In: *arXiv preprint arXiv:2102.11859* (2021).

[86] Yan Yan. *SpConv: Spatially Sparse Convolution Library*. original-date: 2019-01-19T02:57:09Z. Aug. 2022. URL: `https://github.com/traveller59/spconv`.

[87] Jonathan S Yedidia, William Freeman, and Yair Weiss. “Generalized belief propagation”. In: *Advances in neural information processing systems* 13 (2000).

[88] Senthil Yogamani et al. “Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9308–9318.

[89] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *arXiv preprint arXiv:1511.07122* (2015).

[90] Shanshan Zhao et al. “Geometry-aware symmetric domain adaptation for monocular depth estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9788–9798.

[91] Wang Zhao et al. “Towards better generalization: Joint depth-pose learning without posenet”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9151–9161.

[92] Tinghui Zhou et al. “Unsupervised learning of depth and ego-motion from video”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1851–1858.