N-QGNv2: Predicting the optimum quadtree representation of a depth map from a monocular camera

Daniel Braun, Olivier Morel, Cédric Demonceaux, Pascal Vasseur

Pattern Recognition Letters 179 (2024) 94–100

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

N-QGNv2: Predicting the optimum quadtree representation of a depth map from a monocular camera

Daniel Braun [a,*], Olivier Morel [b], Cédric Demonceaux [c,d], Pascal Vasseur [e]

[a] Université de Rouen Normandie, Laboratoire LITIS, Rouen, France
[b] Université de Bourgogne, Laboratoire ImViA, Le Creusot, France
[c] Université de Bourgogne, CNRS UMR 6303 ICB, Dijon, France
[d] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
[e] Université de Picardie Jules Verne, Laboratoire MIS, Amiens, France

ARTICLE INFO

Editor: Daniel Riccio

Keywords:
Deep learning
Quadtree
Compression
Sparse convolutions
Depth map
Monocular image

ABSTRACT

Self-supervised monocular depth prediction is a widely researched field that aims to provide a better scene understanding. However, most existing methods prioritize prediction accuracy over computation cost, which can hinder the deployment of these methods in real-world applications. Our objective is to propose a solution that efficiently compresses the depth map while maintaining a high level of accuracy for navigation purpose. The proposed method is an expansion of the work presented in N-QGN, which utilizes a quadtree representation for compression. This approach has already shown promising results, but we aim to improve it further by making it more accurate, faster, and easier to train. Therefore, we introduce a new method that directly predicts the quadtree structure, resulting in a more consistent prediction, and we revise the network architecture to be lighter and produce state-of-the-art accuracy results, depending on the data compression rate. The new implementation is also faster, making it more suitable for real-time applications. Experiments have been conducted on various scene configuration highlighting the capability of the method to efficiently predicting a reliable quadtree depth representation of the scene at low computation cost and high accuracy.

1. Introduction

The acquisition of depth information by autonomous robots is a crucial aspect of their ability to navigate and understand their environment. The ability to perceive depth allows robots to avoid obstacles, locate objects, and plan paths. However, obtaining accurate depth information is a major challenge in the field of computer vision, as it requires a combination of hardware and algorithms that can handle the complexities of real-world scenes. Despite these difficulties, research in this area continues to advance rapidly, driven by the increasing demand for autonomous systems in a wide range of applications.

The usage of deep learning networks is currently permitting to directly infer depth information from a single RGB image. These methods have been studied for many years and demonstrate accurate results [1–3]. Therefore, one can expect for such methods to come as a replacement of stereo systems. As yet, they are scarcely used for real-time navigation [4]. This limitation can be explained by different factors such as an overly specialized network, a lack of accuracy at long range or the difficulty to be used on embedded systems. Some methods addressed these issues by proposing lightweight solutions for real-time applications [5,6]. But for the most, they 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 [3,7,8].

In our previous work, we introduced N-QGN [9], a quadtree-based depth prediction network that uses submanifold sparse convolutions [10] to generate a quadtree to efficiently compress depth information and reduce network complexity. The results were promising, showing that the network is capable of directly generating a convincing quadtree without the need for full depth information. However, the quadtree structure was not optimal, as it was inferred from partial depth information that had already been predicted. This made it difficult to train correctly, which could lead to some unexpected predictions.

In this paper, we present a novel approach for generating an optimal quadtree representation of depth information. The proposed method includes the following key contributions:

* Corresponding author.
E-mail addresses: daniel.braun@univ-rouen.fr (D. Braun), olivier.morel@u-bourgogne.fr (O. Morel), cedric.demonceaux@u-bourgogne.fr (C. Demonceaux), pascal.vasseur@u-picardie.fr (P. Vasseur).

https://doi.org/10.1016/j.patrec.2024.01.027
Received 23 February 2023; Received in revised form 5 December 2023; Accepted 31 January 2024
Available online 5 February 2024

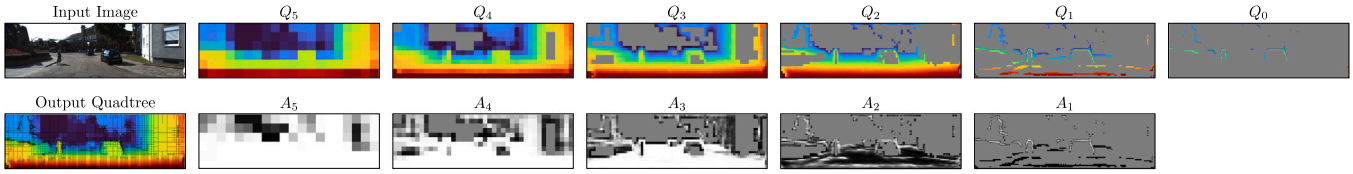

Fig. 1. Quadtree prediction decomposition. On the left is the input image on top and the recomposed quadtree at the bottom. Q_5 to Q_0 are the sparse depth output of each prediction layers. A_5 to A_1 are the subdivision prediction maps, used to define the active sites of the following layers of the decoder.

1. Direct prediction of the quadtree structure by the network, in conjunction with depth prediction.
2. Use of a dense method to guide the training process, ensuring convergence to the optimal solution.
3. A redesigned sparse decoder that is more lightweight while achieving a higher level of accuracy.

The method described in this paper aims to balance the trade-off between prediction accuracy and computation cost by generating an optimum quadtree representation of depth information according to a defined subdivision criterion.

2. Related works

2.1. Monocular depth for navigation

Estimating depth information from a single RGB camera presents an ill-pose problem that was only made possible with the emergence of deep learning. It was demonstrated in [11] its capability by following a training procedure consisting of minimizing the photometric reprojection error between a pair of stereo calibrated images. The method has been improved over the years [12] with the addition of the left–right consistency and the disparity smoothing. Besides, it was demonstrated that similar results could be obtained without the need of a stereo training [2,13].

Regularly, new frameworks are emerging, proposing innovative solutions to outperform the state of the art [3,14,15]. While U-Net architectures [16] with a ResNet 18 encoder produced for some time the most accurate results [7], they have recently been outperformed by the emergence of vision transformers [17]. Yet, the pursuit of high precision is not the sole objective. Some approaches prioritize resilience under diverse weather conditions [18] or focus on constructing efficient architectures for embedded systems [5,6]. Our primary interest lies in implementing the quadtree-based depth estimation method introduced in [9], utilizing quadtree generation to decrease computational costs.

2.2. Octree and quadtree data structure

Hierarchical tree data structures have been widely used in computer vision and navigation to efficiently compress the information. Hornung et al. proposed in [19] to use octree to store large 3D occupancy maps for navigation applications. It permitted the development of frameworks for long-term mapping [20] or real-time 3D mapping [21]. In the same way, quadtree has been used for 2D information compression for real-time navigation application as in [22].

Deep learning methods proposing octree based representation [23] were first introduced and demonstrated the capability of the approach to reconstruct 3D objects from images. Quadtree based applications are derived from octrees and the development of sparse convolution solutions [10]. Chitta et al. proposed QGN in [24] to generate quadtree for segmentation inference and demonstrated the direct gain in term of computational cost without any loss in precision. Upon this approach Braun et al. [9] proposed an adaptation of the precedent framework for depth map compression. As opposed to the segmentation, the depth stores continuous values which cannot be compressed without a loss in the information. The key being to define the most appropriate criterion to limit this loss.

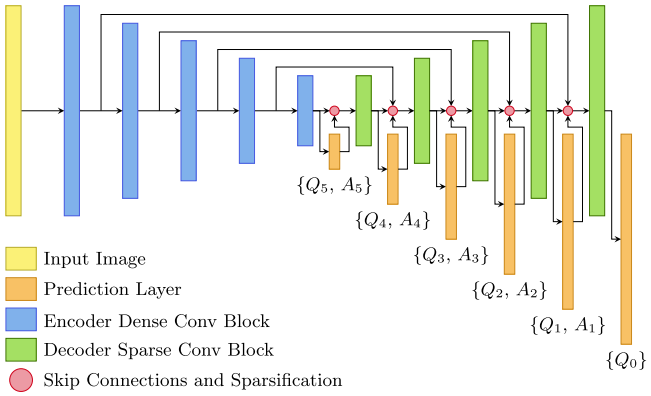

Fig. 2. The network has a U-Net architecture composed of a dense encoder in blue and a sparse decoder in green. The prediction layers are in orange below the decoder. Each one of them outputs a sparse depth prediction (Q_l) and an activation map for the next sparse convolution block (A_l).

3. Method

In this section we present a new framework for jointly optimizing the depth and quadtree data structure prediction. It begins with the definition of the quadtree data structure and its implementation into the network architecture. We then provide details on the training procedure and the use of guidance by a dense depth stereo network, which allows us to achieve state-of-the-art results.

3.1. Quadtree data structure

Quadtrees are principally used for compressing two-dimensional information, like images, because their structure is easily adaptable to the pixel representation. Indeed, a node represents a square area in the image whose dimensions and location are directly related to its depth and position in the tree. Therefore, the quadtree can be defined by $Q = \{l_i, x_i, y_i, d_i \mid i = \{1, .., N\}\}$. It is composed of a set of N nodes, of which the ith is characterized by its depth level l_i, its centroid coordinates (x_i, y_i) and its value d_i, which correspond to the disparity in our application.

In the rest of the paper, the quadtree will be separated by its nodes' depth level and noted $Q_l = \{Q \mid l_i = l\}$. Indeed, each depth level of the quadtree can be seen as a sparse image at the given resolution. The complete quadtree can be seen as a multi-resolution sparse representation of the image as illustrated in Fig. 1.

The data compression within a quadtree structure can be quantified as the ratio between the total pixels present in the initial image and the count of leaf nodes in the quadtree, providing a metric for the degree of data reduction achieved. A high compression ratio implies a highly compressed representation.

3.2. Architecture

3.2.1. Network

The network illustrated in Fig. 2 is based on a U-Net [16] architecture and is composed of a dense encoder and a sparse decoder.

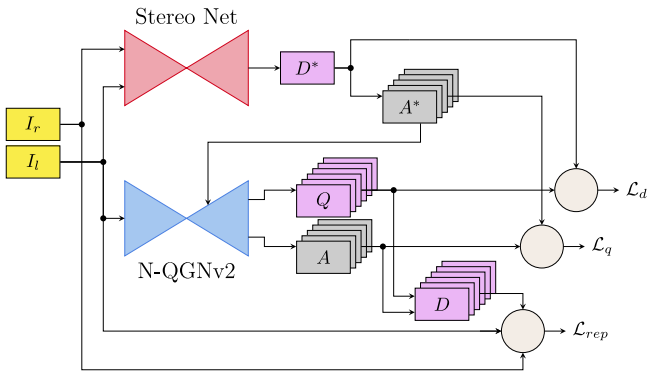

Fig. 3. Training diagram of N-QGNv2. The Quadtree network learning is supervised by stereo network and is also taking advantage of self-supervised monocular training loss function. For more stability, the quadtree subdivision is guided by the reference map during training.

The encoder can be either a ResNet [25], a MobileNetv2 [26] or any other encoder used to predict dense monocular depth. The decoder is a sparse adaptation of the dense equivalent proposed in [2]. It is composed of sparse convolution blocks [10] which allows to only compute information on active sites. The decoder will predict for each layer l the depth Q_l and the set of active sites A_l for the next layer. Therefore, the area to compute in the image will shrink after each layer as the quadtree is getting constructed. Therefore, a large area of the image will not have to be computed while reaching the last layer, which permits to drastically reduce the computation cost.

3.2.2. Quadtree subdivision criterion

The goal of this method is to find the balance between compressing the depth information and the loss of accuracy that results from it. This is done by only keeping important details, such as areas where there is a big difference in values. The method uses the standard deviation of the pixel disparity values in the area and focuses on areas that are close by, which are the most important for navigation. It also sets a limit beyond which depth prediction is not reliable, and so the method does not include details beyond that distance. The rule for deciding which areas to keep is defined as follows:

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

with $p = \{p_i \mid i = \{1, .., N\}\}$ a set of N pixels from the full resolution disparity map representing a square area in the image. 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 ratio of the predicted quadtree. σ is the standard deviation defined by:

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

This subdivision procedure is applied to a full resolution disparity map of reference to construct the optimum quadtree splitting. It results in a set of multiscale binary maps injected to the sparse decoder during training to be used as active sites for layers with the corresponding resolution. At inference time, the network would have learnt how to construct the quadtree, thus the reference binary maps is no longer injected in the decoder.

3.3. Stereo supervision training

3.3.1. Reference stereo network

It is undeniable that inferring disparity from two images produces better results than from one. As demonstrated in [3], it is possible to improve the performances of monocular prediction if the learning is done by distilling the information from a stereo depth network. By following this process, the approach uses as a training reference a self-supervised stereo depth network.

Stereo matching networks such as PSMNet [28] provide high quality depth map prediction, but use very complex architectures requiring significant computing resources. We came up with a lightweight solution based on monocular depth architecture [2] with a stereo input.

3.3.2. Transfert learning

The network is trained in a teacher/student relation as described in Fig. 3. The teacher, or reference, network has been trained upstream to predict dense disparity information from stereo images as presented above. The knowledge of dense depth information allows extracting the optimum quadtree representation fitting to the criterion defined in Section 3.2.2. This dual depth and quadtree subdivision masks, noted respectively d^* and $\mathcal{A}_l^*$, are feeded to the N-QGNv2 network and used as ground truth during training.

3.3.3. Training stability

To stabilize the training, we impose the quadtree partitioning computed by the reference instead of using the one predicted by the network. This prevents the training from diverging or falling into a local minimum. Indeed, since it is a sparse prediction, part of the information is de facto unknown and cannot be evaluated. This guidance ensures the network to be evaluated at the desired locations.

3.3.4. Loss function

As illustrated in Fig. 3, the global loss function is composed of three terms. Jointly, they permit to learn the subdivision probability A_l and sparse disparity Q_l for each l representing the depth level of the quadtree. Both A_l and Q_l are sparse maps and are therefore only evaluated on there active sites, i.e. where the information is present.

The disparity prediction is supervised by the reference disparity map D^* by minimizing the flowing logistic L1 loss function for each Q_l:

$$\mathcal{L}_d(Q_l) = \frac{1}{N_l}\sum_{i=1}^{N_l} \log(|d_i - D_i^*| + 1). \tag{3}$$

The quadtree subdivision probability map A_l aims at minimizing the binary cross entropy function to fit the reference A_l^*:

$$\mathcal{L}_q(A_l) = \frac{-1}{N_l}\sum_{i=1}^{N_l} (a_i^* \log(a_i) + (1 - a_i^*)\log(1 - a_i)). \tag{4}$$

The photometric reprojection loss function (noted $\mathcal{L}_{rep}$) is based on the one presented in [2]. At each prediction scale, the dense depth map is reconstructed from the quadtree sparse prediction using bilinear interpolation.

Ultimately, the global loss function represents the sum of the three previous terms averaged over the number of depth level of the quadtree ($L = 6$ in our current approach). The three terms can be weighted by the coefficients α, β and γ during training. Setting the values to $\alpha = 0.2$, $\beta = 0.8$ and $\gamma = 1$ have shown to perform particularly well.

$$\mathcal{L}_{global} = \frac{1}{L}\sum_{l=0}^{L-1} (\alpha\mathcal{L}_d(Q_l) + \beta\mathcal{L}_q(A_l) + \gamma\mathcal{L}_{rep}(D_l)) \tag{5}$$

3.3.5. Depth pre-training

State of the art monocular methods have demonstrated excellent work at inferring dense depth information. Our method aims at producting a compressed quadtree representation of the depth and is therefore sparse. This specificity affects the training procedure, especially on the last layers of the decoder. Indeed, because of the quadtree subdivision, only a small part of the image is evaluated at this resolution, diminishing the efficiency of the optimization process. To address this problem, the encoder is pretrained by a dense depth prediction network, which is in this instance EPCDepth [7].

Table 1
Depth estimation results on Kitti dataset [27] using the Eigen split evaluation [1]. Results per method are sorted by compression ratio and inference time (fps). For accuracy comparison, dense methods are compressed into quadtree at equivalent compression ratio than quadtree-based N-QGN and N-QGNv2 (ours). **Bold** values are the best score of the category and underlined are the second best.

Methods	Compression ratio	Input resolution	fps↑	Abs Rel↓	Sq Rel↓	RMSE↓	RMSE Log↓	a1↑	a2↑	a3↑
MonoViT [17]	1	192 x 640	6.8	**0.098**	**0.676**	**4.325**	**0.174**	**0.903**	**0.966**	**0.984**
EPCDepth [7]	1	192 x 640	17.6	0.099	0.754	4.490	0.183	0.888	0.963	0.982
Monodepth2 [2]	1	192 x 640	**41.4**	0.108	0.820	4.693	0.188	0.884	0.961	0.981
N-QGN [9]	10	192 x 640	5.8	0.116	0.881	4.946	0.197	0.867	0.955	0.979
MonoViT [17]	10*	192 x 640	6.8*	**0.099**	**0.659**	**4.468**	**0.176**	**0.896**	**0.965**	**0.984**
EPCDepth [7]	10*	192 x 640	17.6*	0.116	0.731	4.624	0.189	0.873	0.962	0.983
monodepth2 [2]	10*	192 x 640	41.4*	0.107	0.756	4.729	0.188	0.879	0.960	0.982
Ours	10	192 x 640	**41.9**	0.110	0.764	4.723	0.188	0.874	0.960	0.982
N-QGN [9]	30	192 x 640	6.0	0.120	0.928	5.084	0.202	0.858	0.951	0.978
MonoViT [17]	30*	192 x 640	6.8*	**0.101**	**0.682**	**4.642**	**0.180**	**0.888**	**0.963**	**0.984**
EPCDepth [7]	30*	192 x 640	17.6*	0.119	0.749	4.784	0.192	0.864	0.960	0.983
monodepth2 [2]	30*	192 x 640	41.4*	0.109	0.763	4.846	0.190	0.873	0.958	0.982
Ours	30	192 x 640	**46.0**	0.113	0.792	4.783	0.189	0.871	0.960	0.983

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

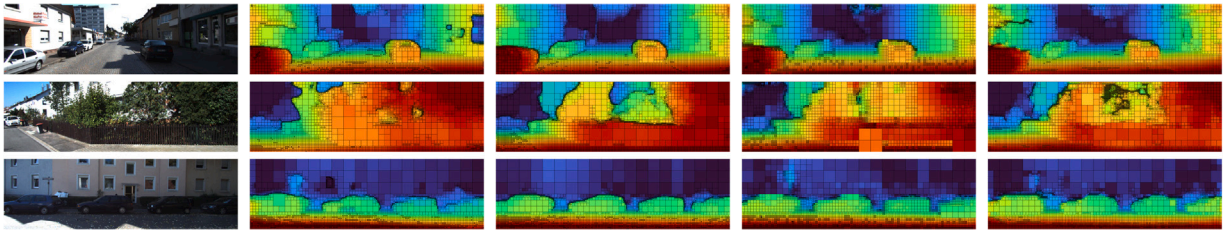

Fig. 4. Qualitative prediction overview. From left to right (a) the input image, (b) the stereo reference map used for training, (c) the Monodepth2 converted into quadtree, (d) the N-QGN method and (e) which is ours. Results are displayed with a compression ratio of 10.

Table 2
Data distribution along the quadtree levels from the root nodes in Q_5 to the leaf nodes in Q_0 at two compression rate. **Bold** values correspond to the highest percentage. The last column represents the quadtree structure likelihood with respect to the stereo reference used for training.

Methods	Comp.	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	L↑
N-QGN	10	7.6%	23.0%	21.6%	**31.8%**	13.1%	2.9%	0.830
MonoViT	10	15.1%	17.6%	20.2%	**33.9%**	9.9%	3.3%	0.881
EPCDepth	10	13.2%	23.0%	25.0%	**28.5%**	7.5%	2.8%	0.844
Monodepth2	10	15.8%	18.7%	19.5%	**33.7%**	9.6%	2.7%	0.878
Ours	10	12.0%	15.6%	21.6%	**36.3%**	11.6%	2.9%	**0.884**
N-QGN	30	18.2%	**32.2%**	22.0%	24.6%	2.4%	0.6%	0.835
MonoViT	30	29.9%	21.0%	**36.4%**	9.3%	2.0%	1.4%	0.906
EPCDepth	30	29.2%	29.4%	**30.8%**	7.6%	2.0%	1.0%	0.882
Monodepth2	30	31.1%	20.5%	**36.5%**	8.9%	1.8%	1.2%	0.903
Ours	30	25.6%	23.3%	**36.9%**	11.6%	1.7%	0.9%	**0.909**

4. Experiments

This paper evaluates the performance of the new N-QGNv2 framework using experiments on the Kitti [27] and CityScapes [29] datasets. The goal is to assess the generated quadtree's quality based on depth prediction and tree structure, and compare its inference speed to other dense methods [2,7] and the previous version of N-QGN [9]. The experiments consider various factors that may impact prediction quality, such as different levels of compression, use of different encoders, and two image resolutions.

4.1. Depth estimation

The goal of the network is to compress depth information efficiently using a quadtree data structure, resulting in a trade-off between speed and accuracy. As presented in Table 1, the performance was evaluated at two compression rates of 10 and 30 and results were compared to state-of-the-art dense methods [2,7,17] and another direct quadtree generation method N-QGN [9]. The new framework improved accuracy and speed compared to N-QGN and showed similar results to dense methods despite a slight decrease in accuracy due to the compression. The recent method MonoViT [17], a transformer based monocular depth estimation, is the one achieving the highest accuracy, but requires extra time to infer prediction.

Our proposed framework showed to be the fastest in terms of inference speed. It is important to note that the time required for converting depth methods to quadtree representation was not considered, making our framework a favorable choice for applications that prioritize processing compressed information.

4.2. Quadtree subdivision

4.2.1. Data distribution

The quadtree compression implies distributing the depth information across different levels of the hierarchical tree. Table 2 compares data distribution of the approach with N-QGN [9] and the three dense

methods EPCDepth [7], monodepth2 [2] and MonoViT [17] for a given compression ratio. The distributions are relatively similar, except for N-QGN, due to its different subdivision criterion. However, all methods have the last level Q_0 representing a small portion of the image, mostly corresponding to edges.

The data distribution provides insights about the scene geometry. The subdivision criterion, which is based on disparity value deviation and applies a threshold to the maximum depth, separates the information in the quadtree. The first levels Q_5 to Q_3 mainly represent flat areas with small local deviations, while the last levels Q_2 to Q_0 mainly represent edges in the image, which are areas with high uncertainty. At high compression rates, these last layers are mostly ignored and only represent a small percentage of the depth map information.

4.2.2. Subdivision prediction

In this approach, the network predicts both the subdivision and the depth. The precision has been evaluated and is oscillating between 85 and 90% of likelihood to the reference quadtree, as presented in the last column of Table 2. It is computed based on the following equation:

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

with $\bar{a}_{ij}$ and a^*_{ij} the jth element of the binary activation maps of respectively the prediction A_i and the reference A^*_i, which defines the quadtree structure. The equation balances the influence of each prediction by considering each A_i equally. A randomly constructed quadtree would score a 50% correspondence with this metric.

These results show the network's ability to predict a coherent quadtree partitioning. Predicting the structure, instead of deducing it as in previous work, improves the results. The method with the highest compression rate slightly improves the likelihood, due to its compressed information resulting in mostly zero values in the last layer predictions, which corresponds to the area of highest uncertainty.

4.3. Qualitative results

The results of the new framework are compared to the reference map, monodepth2, and N-QGN in Fig. 4. The EPCDepth method is not included due to the higher quadtree structure likelihood achieved by monodepth2, as shown in Table 2. The black lines in the figure represent the borders of each node to enhance readability of the quadtree structure.

Three images are presented to visualize the performances of the method under different scenarios. Some parts of the scene, such as the road, cannot be highly compressed due to the geometry and presence of a vanishing point. On the other hand, other parts such as obstacles or fronto-parallel surfaces can be highly compressed as they have low deviations in values.

The predicted quadtree structure by our approach offers a visually coherent representation with respect to the reference map. Some inconsistencies can still be noted on bushes in the second image, as it seems to be the case for most methods. Compared to N-QGN, our prediction is smoother and less prone to over-subdivising nodes. Ultimately, it is challenging to determine whether the result was directly generated by a Quadtree Generating Network or constructed from dense information.

4.4. Runtime

The runtime experiments have been conducted on a computer equipped with a GPU Nvidia Quadro P620 and a process Intel Core i7-9850H CPU at 2.60 GHz × 12. Although this setup may not reflect the conditions of embedded systems, it provides a basis for comparison between dense and quadtree-based methods. Part of the results were presented in Table 1 along side the corresponding accuracy metrics. The study has been extended to more scenarios to examine the runtime based on the choice of encoder and input image size, as depicted in Fig. 5.

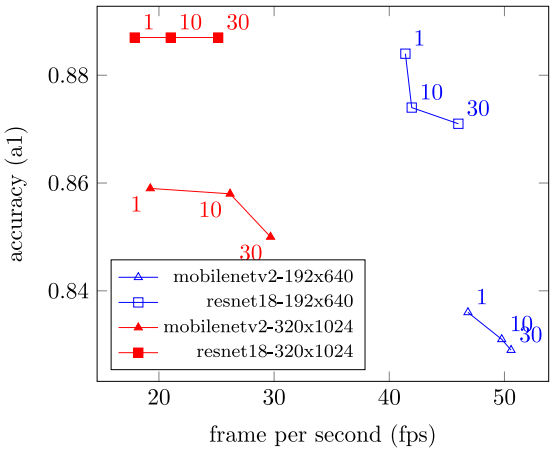

Fig. 5. Runtime with respect to the encoder, compression ratio and input image resolution. The label values on the point represent the compression ratio.

Table 3
Results on the CityScapes dataset [29].

Methods	Comp.	Abs Rel↓	Sq Rel↓	RMSE Log↓	a1↑	a2↑	a3↑
MonoViT	1	**0.207**	**2.297**	**0.297**	**0.622**	**0.878**	**0.954**
EPCDepth	1	0.322	4.393	0.451	0.447	0.739	0.873
Monodepth2	1	0.212	2.749	0.323	0.628	0.849	0.935
Ours	17	0.250	3.041	0.359	0.529	0.824	0.924

Comparison with N-QGN [9] is not included in the Fig. 5, as it was demonstrated in Table 1 the new framework is faster. This is due to the improvements made to both the architecture and implementation. The new framework predicts the subdivision of nodes in the quadtree, which is faster than the previous solution that calculated it geometrically from partial data. Besides, the framework has been implemented with the *SpConv* library [30], which provides a highly optimized sparse convolution solution and is faster than the previously used *SparseConvNet* [10]. The dense-to-sparse operation was also optimized, as it previously ran slow due to the re-indexing procedure.

Fig. 5 highlights the trade-off associated with quadtree solutions, which offer faster inference at the cost of lower accuracy. The speed gains from solutions with a ResNet18 encoder are minimal at low resolution. With a QGN architecture, the speed gain is solely attributed to the sparse decoder, as the encoder remains unchanged compared to dense methods. Therefore, to evaluate the impact of the encoder on inference runtime, the lighter Mobilenetv2 [26] was used. As expected, the inference was faster, reaching 50 fps with our framework at low resolution, but at the cost of a drop in accuracy.

4.5. CityScapes dataset evaluation

It is proposed to extend the study to the CityScapes dataset [29], which contains high-resolution images of urban street scenes captured in various cities across Germany. To assess the robustness of the approach, it will not be fine-tuned on the CityScapes data but rather be directly evaluated using the pre-trained weights from the Kitti dataset [27]. The results presented in Table 3 demonstrate the capability of the method to achieve results comparable to CNN dense approaches. In this case, the compression ratio has little impact on the depth prediction accuracy, with our compressed method yielding better results than the dense EPCDepth [7] on this particular dataset. MonoViT [17] stands out for delivering superior results, leveraging a vision transformer architecture, a recognized outperformer compared

to CNN-based models. This outcome sparks discussions regarding the potential development of a quadtree generating network tailored for transformers.

It can be noted, the compression rate did not remain consistent from one dataset to another, with the new results being twice as compressed as those obtained on the Kitti dataset. This can be explained to changes in scene geometry, as the compression rate is closely linked to it. Similarly, a change in camera orientation, or focal length, may also impact the compression rate by allowing for better compression of particular areas such as the ground.

5. Conclusions

In this paper, we presented a method to efficiently 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 similar method N-QGN, 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 reinforced the potential of quadtree generating networks and the benefits of using sparse convolutions to reduce computation cost without significant loss of accuracy. Even if the gain is slight at low resolution, it becomes genuinely interesting with bigger input images. The network's capability to efficiently predict quadtree node subdivisions opens up possibilities for more advanced subdivision criteria, such as taking into account the semantic information of the scene to determine depth resolution based on class.

Comparative analysis with transformer networks showcased our method's strength in faster inference. However, the distinct accuracy gap prompts contemplation about its compatibility with this novel architectural setting. While this transition is not straightforward, we hold strong confidence in the feasibility of such transformation.

CRediT authorship contribution statement

Daniel Braun: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Software, Visualization, Writing – original draft, Writing – review & editing. **Olivier Morel:** Supervision, Validation, Writing – review & editing. **Cédric Demonceaux:** Funding acquisition, Supervision, Validation, Writing – review & editing, Project administration. **Pascal Vasseur:** Funding acquisition, Project administration, Supervision, Validation, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgment

This work is supported by the ANR CLARA project, grant ANR-18-CE33-0004, of the French Agence Nationale de la Recherche (National Research Agency). We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Daniel BRAUN reports financial support was provided by ANR CLARA project, grant ANR-18-CE33-0004 of the French Agence Nationale de la Recherche (National Research Agency).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patrec.2024.01.027.

References

[1] D. Eigen, R. 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.

[2] C. Godard, O. Mac Aodha, G.J. Brostow, Digging into self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3828–3838.

[3] Z. Chen, Coauthors, 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. 15 529–15 538.

[4] X. Dong, M.A. Garratt, H.A. Abbass, Towards real-time monocular depth estimation for robotics: A survey, 2021, arXiv preprint arXiv:2111.08600.

[5] L. Wang, M. Famouri, A. Wong, Depthnet nano: A highly compact self-normalizing neural network for monocular depth estimation, 2020, arXiv preprint arXiv:2004.08008.

[6] M.-J. Chiu, W.-C. Chiu, J.-H. Chuang, 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.

[7] R. Peng, Y. Wang, R. Lai, Y. Cai, Excavating the potential capacity of self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 15 560–15 569.

[8] S.M.H. Miangoleh, L. Dille, S. Mai, S. Paris, Y. Aksoy, 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, 2021, pp. 9685–9694.

[9] D. Braun, O. Morel, C. Demonceaux, N-qgn: Navigation map from a monocular camera using quadtree generating networks, in: 2022 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2022.

[10] B. Graham, M. Engelcke, L. van der Maaten, 3D semantic segmentation with submanifold sparse convolutional networks, in: CVPR, 2018.

[11] R. Garg, V.K. Bg, I. Reid, Unsupervised cnn for single view depth estimation: Geometry to the rescue, in: European Conference on Computer Vision, Springer, 2016, pp. 740–756.

[12] C. Godard, O. Mac Aodha, G.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.

[13] T. Zhou, M. Brown, D.G. Lowe, 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.

[14] M. Poggi, F. Aleotti, S. Mattoccia, On the uncertainty of self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3227–3237.

[15] V. Guizilini, S. Ambrus, A. Pillai, R. Raventos, A. Gaidon, 3D packing for self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2485–2494.

[16] O. Ronneberger, P. Fischer, T. 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.

[17] C. Zhao, Coauthors, Monovit: Self-supervised monocular depth estimation with a vision transformer, in: International Conference on 3D Vision, 2022.

[18] K. Saunders, G. Vogiatzis, L.J. Manso, Self-supervised monocular depth estimation: Let's talk about the weather, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 8907–8917.

[19] A. Hornung, M. Wurm, C. Bennewitz, K.M. Stachniss, W. Burgard, Octomap: An efficient probabilistic 3D mapping framework based on octrees, Autonomous Robots 34 (3) (2013) 189–206.

[20] L. Sun, A. Yan, C. Zaganidis, Z. Zhao, T. Duckett, Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-D-LiDAR data, IEEE Robot. Autom. Lett. 3 (4) (2018) 3749–3756.

[21] S. Vanneste, B. Bellekens, M. Weyn, 3DVFH+: Real-time three-dimensional obstacle avoidance using an octomap, in: MORSE 2014 Model-Driven Robot Software Engineering: proceedings of the 1st International Workshop on Model-Driven Robot Software Engineering co-located with International Conference on Software Technologies: Applications and Foundations, STAF 2014, York, UK, July 21 2014/Assmann, Uwe [edit.], (1319), 2014, pp. 91–102.

[22] K. Wang, W. Ding, S. 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.

[23] M. Tatarchenko, A. Dosovitskiy, T. 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.

[24] K. Chitta, J.M. Alvarez, M. 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.
[25] K. He, X. Zhang, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[26] M. Sandler, M. Howard, A. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[27] A. Geiger, P. Lenz, R. 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.
[28] J.-R. Chang, Y.-S. Chen, Pyramid stereo matching network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5410–5418.
[29] M. Cordts, Coauthors, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.
[30] Y. Yan, SpConv: Spatially sparse convolution library, 2022, URL https://github.com/traveller59/spconv, original-date: 2019-01-19T02:57:09Z.