Deep Line Encoding for Monocular 3D Object Detection and Depth Prediction

Ce Liu[1]
ce.liu@vision.ee.ethz.ch
Shuhang Gu[2]
shuhanggu@gmail.com
Luc Van Gool[1,3]
vangool@vision.ee.ethz.ch
Radu Timofte[1,4]
radu.timofte@vision.ee.ethz.ch

[1] Computer Vision Lab
ETH Zurich
Switzerland
[2] University of Sydney
Australia
[3] ESAT
KU Leuven
Belgium
[4] University of Würzburg
Germany

Abstract

In autonomous driving scenarios, straight lines and vanishing points are important cues for single-image depth perception. In this paper, we propose the deep line encoding to make better use of the line information in scenes. More specifically, we transform potential lines into parameter space through the deep Hough transform. The aggregation of features along a line encodes the semantics of the entire line, whereas the voting location indicates the algebraic parameters. For efficiency, we further propose the novel line pooling to select and encode the most important lines in scenes. With deep line encoding, we advance the state-of-the-art on KITTI single-image 3D object detection and depth prediction benchmarks. The code is available at https://github.com/cnexah/DeepLineEncoding.

1 Introduction

Recovering depth from RGB images has been a long-standing problem in vision and robotics. For example, a key step in 3D object detection is to predict the distance (depth) of foreground objects. Whereas in 3D reconstruction, the depth of all pixels is often required. It is known that for a single RGB image, in general, the problem is ill-posed. Thereby the dominant approaches resort to multiple images from different viewpoints and locate points in 3D space by triangulation [15]. However, the human being can make a rough estimate of depth even from a single eye. The observation motivates researchers to explore various cues for single-image depth perception, such as shading [40], defocus [13], and so on.

Restricted to street scenes, Dijk and Croon [7] show an important cue to infer objects' depth, *i.e.*, objects that are further away tend to appear higher in the image. However, bringing into full play the prior for accurate depth perception is by no means trivial. Because the road surface has been assumed to be a rough plane, which is often violated in reality. For example, there are usually different slopes in different areas, and cars might even be on curb bricks. In those cases, the simplified relation between objects' depth and their image coordinates might deviate from the truth. How to model the above complex situation effectively is still an open question.

In this paper, we propose to exploit the basic primitives, like straight lines and vanishing points, in man-made environments, especially in autonomous driving scenarios. Because their angle or position indicates the slope of the road and even the 3D layout of the whole scene. To explicitly represent the semantics (*e.g.* guard rail or horizontal line) and algebraic parameters of lines, we perform deep Hough transform [8, 25] on the feature map of deep networks. The voting for a line is obtained by aggregating the features along the line, which encodes the semantic information from the entire line. In addition, the angle and position are indicated by the voting location in parameter space.

In parameter space, feature maps are sparser than the ones in image space in the sense that most of the locations are meaningless. Only a few elements represent straight lines that make sense, whereas the vast majority correspond to random aggregation. For efficiency, we propose the line pooling module to select important lines in parameter space. The lines are finally represented as a vector, and can be easily incorporated into conventional networks.

We apply our design to off-the-shelf frameworks for monocular 3D object detection and depth prediction. The main challenges in the two tasks are to predict the distance (depth) of foreground objects and estimate the dense depth map respectively. With deep line encoding, we advance the state-of-the-art on KITTI monocular 3D object detection and depth prediction benchmarks [11]. The improvements demonstrate the effectiveness of our design.

In summary, our main contributions are as follows: (1) We introduce the line information in scenes as a novel cue for single-image depth perception. (2) We propose a novel architecture to exploit the line information, which fits well into off-the-shelf frameworks. (3) We advance the state-of-the-art on KITTI single-image 3D object detection and depth prediction benchmarks.

2 Related Work

In this section, we briefly review recent advances in monocular 3D object detection and depth prediction.

2.1 Monocular 3D Object Detection

In monocular 3D object detection, a series of attributes of the object is required to estimate, including the 3D coordinate, size and orientation. However, our method only aims to help the network better estimate the Z coordinate (depth) of the object. Thereby we divide existing methods into three main categories according to their way to infer the depth of objects. It's noteworthy that, compared with the dense depth estimation task, only the depth of foreground objects are required. Thereby more priors can be exploited, and the methods can be more sophisticated.

The most popular way is to make use of the depth information provided by external models. A typical example of the external model is DORN [10], which learns to convert images to dense depth maps. Pseudo-LiDAR [36] back-projects the depth map into 3D points and then apply off-the-shelf LiDAR-based 3D object detectors. PatchNet [29] encodes camera calibration information by spatial coordinates transformation. However, instead of further improving depth estimation accuracy, such approaches focus more on making better use of the prediction results from existing models.

Deep3DBox [30] and RTM3D [22] infer depth through perspective relationships between 3D corners and their 2D projections. The motivation is to explicitly encode the geometric prior that objects that are further away appear smaller. However, other cues such as object's image position may be ignored. In addition, prediction error in orientation, dimension, and 2D bounding box will harm the accuracy of depth estimation.

There are also works that directly regress the depth of centers of foreground objects. Although simple, many works have demonstrated the effectiveness of the paradigm. More specifically, M3D-RPN [2] designed a novel depth-aware convolutional layer which is able to learn spatially-aware features. MonoDIS [33] introduced the disentangle transformation to isolate the contribution of groups of parameters to a given loss. VisualDet3D [27] proposed the so called ground-aware convolution to incorporate features from contacting points between objects and the ground plane.

2.2 Monocular Depth Prediction

Focusing on fully-supervised methods, we find that recent works mainly proceed in two directions.

One line is to design more sophisticated loss functions. Eigen *et al.* [9] proposed the scale-invariant loss to save the network from learning the absolute global scale. DORN [10] and SORD [6] treat depth network learning as an ordinal regression problem. Jiao *et al.* [19] investigated the long tail property of depth values and proposed the attention-driven loss. Wei *et al.* [38] formed a high-order geometric constraint called virtual normal, biasing the network to produce depth maps with better surface. TSN [35] and CTN [31] jointly predict depth, surface normal and semantic segmentation to exploit cross-task affinity patterns.

Proposing novel network architectures is the other direction. Aich *et al.* [1] and Xu *et al.* [37] fuse the feature maps from different stages of the backbone by the bidirectional attention modules and continuous graphical models respectively. Lee *et al.* [21] combine multi-scale depth map candidates in the Fourier domain. Chen *et al.* [3] proposed the spatial attention blocks to guide the network attention to global structures or local details across different feature layers. Huynh *et al.* [18] developed the depth-attention volume to exploit planar structures in the scene.

3 Approach

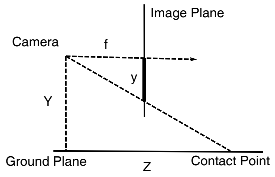
In this section, we start with an analysis of the simplified projection model. Its limitation motivates us to explicitly encode the line information from the scene. As an effective way, the Hough transform is briefly reviewed, and then we take a step further by proposing the novel line pooling module. At last, we present the overall architecture with deep line encoding.

3.1 Depth from Lines

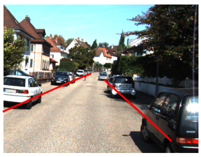
In autonomous driving scenarios, an important cue to depth prediction is object's vertical position in the image. As shown in Figure 1 (a), in the ideal case where the ground plane is perfectly horizontal, the distance of the object can be easily obtained by:

$$Z = \frac{fY}{y}, \tag{1}$$

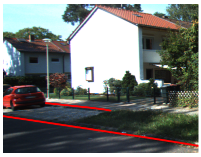
where f and Y are camera's focal length and height respectively, and y is the image vertical coordinate difference between the principal point and the 2D projection of object's ground contact point.



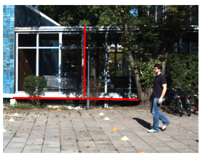
(a) Projection Model



(b) Slope



(c) Step



(d) Pose Variation

Figure 1: (a) Illustration of depth estimation from the image coordinate of the object. (b)-(d) Examples of real situations. We highlight some representative lines (red) that provide information about slope, step and camera pose variation respectively.

In real-world applications, the road is rarely a perfect plane, and the projection relation can be affected by various factors, such as slope, step, camera pose variation, and so on. Special structures in the scene, especially straight lines and vanishing points, can indicate the geometric layout of the scene and help the convolutional networks reason the real projection relation, as shown in Figure 1 (b)-(d).

3.2 Deep Line Encoding

In this section, we introduce the deep line encoding to make better use of the line information from the scenes.

3.2.1 Hough Transform

The traditional Hough transform algorithm [8] usually takes a binary edge map as input. A straight line l is represented by a point (θ, ρ) in the parameter space, where θ is the angle between the x-axis and the normal vector of the line, and ρ is the distance from the origin to the line. Its power is limited because the single point (θ, ρ) is short of semantic context and prone to noise.

Recently, Lin *et al.* [25] and Han *et al.* [14] proposed the deep Hough transform. The input is replaced with features from deep networks, which enables end-to-end training and is more robust. Given a feature map X, the transformed map Y is calculated by the following equation:

$$Y(\theta, \rho) = \sum_{(x,y) \in l} X(x,y), \tag{2}$$

where l is the line parameterized by (θ, ρ).

It's known that in last convolutional blocks of deep networks, the feature maps are rich in semantics. Thereby the aggregation of features along the line can be an encoding for the semantics of the entire line, which is necessary for the line pooling module to select the most important lines. The voting position (θ,ρ) in the transformed map Y indicates the algebraic parameters of the line.

In implementation, we set the origin to be the center of the feature map X, and discretzie $\theta \in [0, 180^\circ)$ and $\rho \in [-\sqrt{W^2+H^2}/2, \sqrt{W^2+H^2}/2]$ into bins, where W and H are the width and height of the feature map X respectively.

3.2.2 Line Pooling

In deep Hough transform, the size of the transformed map Y is usually large but most of the elements are invalid aggregation. As only a few elements of Y represent lines that make sense, we propose a line pooling module to compress Y. The pipeline of our proposed line pooling module is shown in Figure 2 (a).

As introduced in the previous section, the feature vector $Y(\theta,\rho)$ aggregates features in X along the corresponding line l, and encodes the semantics of the entire line. However, $Y(\theta,\rho)$ itself is ignorant of its own absolute position in Y, that is, the algebraic parameters of l. Thereby we first fuse Y with the coordinate map [26] by convolutional layers. The coordinate map indicates the position by filling in the location (i, j) of the coordinate map with vector $[i, j]$. Denoting the resulting feature map as Z, the vector $Z(\theta,\rho)$ is expected to contain not only semantic but also algebraic information about the corresponding line l.

The next step is to select meaningful lines in Z. To do so, we first apply stacks of convolutional layers on Z to generate a score map. The score in each location indicates the importance of the corresponding line. Then we normalize the scores in the map through the soft-max operation. The resulting probability map M represents the probability of selection.

Finally, the expectation, *i.e.*, channel-wise sum of $Z \cdot M$, is regarded as the output line vector. It means the weighted sum of information from all the possible lines. In implementation, we divide channels of Z into k groups, and predict M with k channels. We perform group-wise pooling to select multiple lines at the same time.

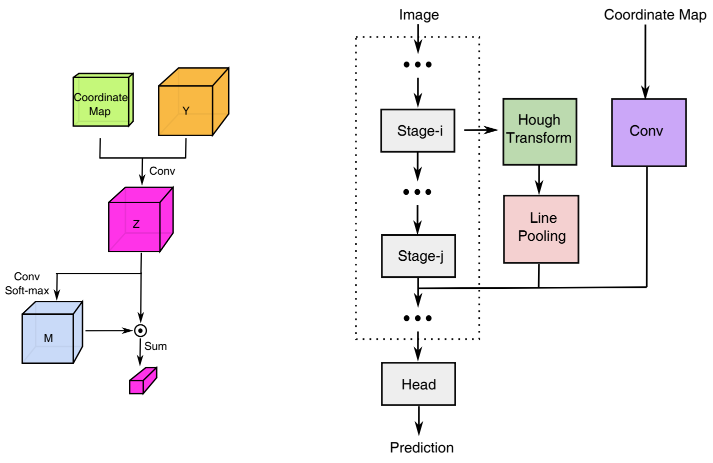
Comparing with max or average pooling, the line pooling module is more flexible. In experiment we found the probability distribution in M might concentrate on a single global maximum, or several local maxima, or even uniformly distribute around all locations.

3.3 Overall Architecture

Our method is generic and can be applied to various frameworks. For generality, we suppose the rough architecture for monocular 3D object detection or depth prediction to be as shown in Figure 2 (b). Given an input image, the backbone extracts features at first, and then the head makes prediction for the specific task.

In most cases, such as in VGG [34] and ResNet [16], the backbone is composed of a series of stages. The feature maps from early stages are of high resolution but semantically weak. After stacks of convolutional and pooling layers, features from different locations are hierarchically aggregated in a complex way. Thereby in later stages the feature maps are semantically stronger but of lower resolution.

We perform deep Hough transform [14, 25] on feature maps from an early stage of i. The choice is natural since high resolution maps preserve more accurate location information. A key step in deep Hough transform [14, 25] is to aggregate the features along the line. The resulting vector can be viewed as an explicit representation for the semantics of the entire line, which is necessary for the line pooling module to distinguish between the guard rail and the horizontal line. The angle and position of the line are indicated by the voting location.



(a) Line Pooling (b) Overall Architecture

Figure 2: (a) Pipeline of the line pooling module. The circle with black dot denotes element-wise product. (b) Overview of the architecture with deep line encoding. Dotted box indicates the backbone.

After deep Hough transform [14, 25], key lines in the scene will be selected by the line pooling module and encoded as a short vector.

To incorporate the line information into the backbone, we concatenate the feature map F from the stage of j with the line vector. To do so, we up-sample the line vector to the same resolution as F in advance. The relative position of the object with respect to the lines or the principle point is an important factor, thereby we also append the coordinate map [26]. We concatenate F, the line vector and the coordinate map and compress into the same number of channels as F. The resulting feature map will replace F and be fed into the stage of $j+1$.

While the deep line encoding module is plugged into the backbone, other parts of the framework are kept the same as original. We train the framework in an end-to-end manner, without any extra supervision than the common monocular 3D object detection or depth estimation frameworks. In experiment (Section 4.4), we found the network can discover related lines automatically.

4 Experiment

In this section, we analyze and verify each component in deep line encoding by detailed ablation study and visualization. We also apply deep line encoding to the state-of-the-art frameworks for monocular 3D object detection and depth prediction.

4.1 Monocular 3D Object Detection

Dataset The KITTI 3D Object Detection dataset [11] consists of 7,481 training images and 7,518 testing images from autonomous driving scenes. Following Chen *et al.* [4], we split the training data into 3,712 training and 3,769 validation images. As has been the focus of prior work [2], we primarily compare methods using the car class.

Configuration	Coordinate	Line Vector	AP_{3D}		
			Easy	Moderate	Hard
a			23.63	16.16	12.06
b	✓		25.21	16.48	12.46
c		✓	24.92	16.41	12.33
d	✓	✓	**26.49**	**16.75**	**13.07**

Table 1: Different configurations ablation study. Configuration a is from Liu *et al.* [27].

VisualDet3D We apply deep line encoding to the state-of-the-art monocular 3D object detector, *i.e.*, VisualDet3D [27]. It is a single end-to-end network composed of a backbone and two task-specific heads. The backbone is responsible for extracting feature maps over the input image. The default is to take the first three stages of ResNet-101 [16] as backbone. The first head performs convolutional object classification; the second head performs convolutional 3D bounding box regression. The 3D bounding box is disentangled into a group of parameters to be regressed, including 2D projection location of the 3D center, depth, size and orientation. The detector is trained with focal loss [24] for the classification branch and smooth l1 loss [12] for the regression branch.
Line Encoding We choose feature maps from the first stage to perform deep Hough transform. The transformed map will be compressed into a short vector by the line pooling module. The line vector and coordinate map will be fused with the feature map from the second stage and then fed into the third stage.
Training Details We adopt the Adam algorithm [20] to optimize network parameters for 40 epochs. We use an initial learning rate of 1e-4, a cosine annealing scheduler [28] with target learning rate of 1e-5, a batch size of 8, and no weight decay. The data pre-processing and augmentation are exactly the same as VisualDet3D [27]. We increase the loss weight of depth prediction branch from 3.0 to 5.0, while other hyper-parameters are kept the same. For deep Hough transform [14, 25], we set the resolution of ρ and θ to be 3 pixel units and 3°.

4.2 Ablation Study

We perform detailed ablation study on the validation set. The experiments are repeated 5 times for average. We first evaluate the effects of critical components in deep line encoding, including the coordinate map and the line vector. Main results are shown in Table 1. More details are presented in supplementary. From Table 1 (b), we observe that simply adding the coordinate map to the framework can boost the performance significantly. The result coincides with the conclusion of Liu *et al.* [26], *i.e.*, the vanilla network might fail to learn the absolute coordinate information. Table 1 (c) shows the improvement from the line vector alone, indicating that the information about straight lines in scenes is beneficial for depth perception. We observe a further improvement from Table 1 (d) by combining the coordinate map with the line vector.

Then we evaluate the effects of each component in the line pooling module. The results are shown in Table 2. One of the most important design is the coordinate map, which represents the algebraic parameters of lines. If we remove the coordinate map, there will be a drop in performance, especially in the hard case of AP_{3D}. The other key design is the soft-max operation. We verify its effects by directly selecting the average or maximum in each channel of Z. The results show that the soft-max operation is better. Particularly, selecting the maximum even harms the performance.

We further apply deep line encoding to M3D-RPN [2] to demonstrate the generalization

Pooling	AP_{BEV}			AP_{3D}		
	Easy	Moderate	Hard	Easy	Moderate	Hard
LinePooling	**34.06**	**22.59**	**16.96**	**26.43**	**16.72**	**13.02**
-Coordinate	33.25	22.11	16.77	24.96	16.44	12.52
-Soft-max (avg)	33.34	22.55	16.91	24.77	16.26	12.55
-Soft-max (max)	31.50	21.68	16.29	23.28	15.80	11.75

Table 2: Pooling ablation study.

Method	AP_{BEV}			AP_{3D}		
	Easy	Moderate	Hard	Easy	Moderate	Hard
M3D-RPN [2]	20.85	15.62	11.88	14.53	11.07	8.65
+ **Line Encoding (ours)**	**22.97**	**16.59**	**13.33**	**16.36**	**11.93**	**9.35**
VisualDet3D [27]	29.70	20.98	16.20	23.63	16.16	12.06
Line Encoding (ours)	**34.06**	**22.59**	**16.96**	**26.43**	**16.72**	**13.02**

Table 3: Frameworks ablation study.

ability across frameworks. M3D-RPN [2] is a simple single-stage network composed of a backbone and two task-specific heads. It takes DenseNet-121 [17] as backbone, while removing the final pooling layer and dilating each convolutional layer in the last Dense-Block by a factor of 2. The heads are used for object classification and attributes regression respectively. We adopt the configuration without depth-aware convolutional layers and optimize the model for 100 thousands iterations directly.

We select the feature map from the second Dense-Block to extract the line vector. Then together with the coordinate map, we fuse the line vector with the original feature map and feed into following modules. As shown in Table 3, deep line encoding module improves the performance of M3D-RPN [2] under all the metrics. We also list the performance of VisualDet3D [27] for comparison.

4.3 Computation Cost

Model size and test time are shown in Table 4. Following Lin *et al.* [25], the deep Hough transform is implemented as matrix multiplication. We compress the feature maps into 16 channels before feeding into the deep Hough transform module. The accumulator is then expanded to 256 channels through convolutional layers. Thereby the line vector has a length of 256. We evaluate the test time for the two models both on NVIDIA Tesla V-100 GPU with a batch size of 1.

Method	Model Size (MB)	Test Time (s/image)
VisualDet3D [27]	55.68	0.053
+ **Line Encoding (ours)**	61.84	0.060

Table 4: Model size and processing time.

4.4 Visualization of Lines

We perform inverse Hough transform on the probability map M to visualize the lines selected by the line pooling module. The results are shown in Figure 3. The first column shows the input image. The second column shows an example of the feature map to perform deep

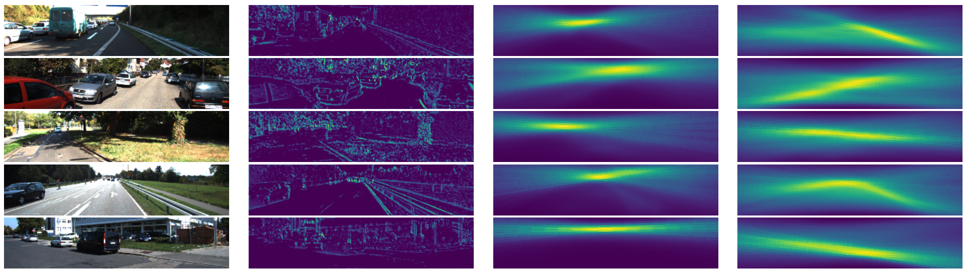


Figure 3: Visualization of lines from the line pooling module.

Hough transform. The third and the forth columns show the inverse Hough transform of different channels of M.

In Figure 3 (c) and (d), the bright areas show the distribution of important lines generated by different channels of M. The areas are blur because the probability is often distributed among a combination of multiple lines. However, we still observe that the selected lines often align with special structures in scenes, such as guard rails, horizontal lines, vanishing points and so on. Specifically, Figure 3 (c) focuses on vanishing points or horizontal lines, while Figure 3 (d) is sensitive to the guard rails. Although we select only a single value from each channel, the soft-max operation is flexible such that the probability can concentrate on a single global maximum, or several local maxima. On Figure 3 (d) it is obvious that both the left and right guard rails are selected.

4.5 Comparison with State-of-The-Art Methods

Table 5 presents the performance of recent methods on the KITTI monocular 3D object detection benchmark [11]. Our method shows superiority over VisualDet3D [27] and achieves the state-of-the-art on easy and moderate cases. Particularly, on the easy case of AP_{3D} our method increases by 2.58 points of AP (24.23 *vs*. 21.65). For the hard case, our method lags behind MonoPair [6]. However, our method does not exploit many popular improvements, such as the feature pyramid network [23] and deep layer aggregation [39], which are helpful for small objects. These improvements are complementary to deep line encoding and should boost the accuracy of hard case further.

Method	AP_{BEV}			AP_{3D}		
	Easy	Moderate	Hard	Easy	Moderate	Hard
M3D-RPN [2]	21.02	13.67	10.23	14.76	9.71	7.42
MonoPair [6]	24.12	18.17	**15.76**	16.28	12.30	**10.42**
RTM3D [22]	19.17	14.20	11.99	14.41	10.34	8.77
PatchNet [29]	22.97	16.86	14.97	15.68	11.12	10.17
VisualDet3D [27]	29.81	17.98	13.08	21.65	13.25	9.91
+ Line Encoding (ours)	**31.09**	**19.05**	14.13	**24.23**	**14.33**	10.30

Table 5: Performance comparison on KITTI monocular 3D object detection benchmark.

Method	SILog	sqErrorRel	absErrorRel	iRMSE
DORN [10]	11.77	2.23	**8.78**	12.98
VNL [38]	12.65	2.46	10.15	13.02
SORD [6]	12.39	2.49	10.10	13.48
BANet [1]	**11.55**	2.31	9.34	**12.17**
GAC [27]	12.13	2.61	9.41	12.65
+ **Line Encoding (ours)**	11.81	**2.22**	9.09	12.49

Table 6: Performance comparison on KITTI single-image depth prediction benchmark.

4.6 Monocular Depth Prediction

Dataset The KITTI monocular depth prediction dataset [11] consists of 42,949 training images, 1,000 validation images, and 500 test images, annotated with sparse point clouds.
GAC We select the open-source framework GAC [27] to evaluate deep line encoding. GAC is based on the U-Net structure [32], and is composed of a backbone and a head. The default is to take the ResNet-34 [16] as backbone. The head is to fuse features from different stages into a feature map with high resolution and rich semantic information, and then perform convolutional depth regression. The network is trained with a scale-invariant loss [9] and a smoothness loss.
Line Encoding We perform deep Hough transform [14, 25] on the features from the second stage of the backbone. The line vector will be fused with coordinate maps and feature maps from the third stage.
Training Details We adopt the Adam algorithm [20] to optimize network parameters for 8 epochs. We use an initial learning rate of 1e-4, a cosine annealing scheduler [28] with target learning rate of 1e-5, a batch size of 8, and no weight decay. For deep Hough transform [14, 25], we set the resolution of ρ and θ to be 3 pixel units and 3°.
Performance on KITTI As shown in Table 6, with deep line encoding, we achieve a better performance than GAC [27] under all the metrics. Especially in sqErrorRel, we observe a significant improvement (2.22 *vs*. 2.61). Comparing with other published methods, we achieve the state-of-the-art in terms of the metric of sqErrorRel, and the second best rating under absErrorRel and iRMSE. Qualitative results and validation set performance are presented in supplementary.

5 Conclusion

Recovering depth from a single RGB image is a challenging task. In this paper, we have shown that line structures in scenes provide valuable information for depth perception. Furthermore, we presented a simple architecture to exploit the lines inside ConvNets. Our method obtains state-of-the-art results on single-image 3D object detection and depth prediction benchmarks. Finally, our study suggests that despite the success of deep ConvNets, it is still necessary to incorporate prior knowledge and design more efficient representation for the specific task.

6 Acknowledgment

This paper was supported by the China Scholarship Council and a Microsoft Azure grant.

References

[1] Shubhra Aich, Jean Marie Uwabeza Vianney, Md Amirul Islam, Mannat Kaur, and Bingbing Liu. Bidirectional attention network for monocular depth estimation. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[2] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3D region proposal network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9296, 2019.

[3] Tian Chen, Shijie An, Yuan Zhang, Chongyang Ma, Huayan Wang, Xiaoyan Guo, and Wen Zheng. Improving monocular depth estimation by leveraging structural awareness and complementary datasets. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 90–108, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58568-6.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3D object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3D object detection using pairwise spatial relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[6] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.

[7] Tom van Dijk and Guido de Croon. How do neural networks see depth in single images? In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[8] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972. ISSN 0001-0782. doi: 10.1145/361237.361242.

[9] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[10] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[11] 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*, pages 3354–3361. IEEE, 2012.

[12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[13] Shir Gur and Lior Wolf. Single image depth estimation trained via depth from defocus cues. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7683–7692, 2019.

[14] Qi Han, Kai Zhao, Jun Xu, and Ming-Ming Cheng. Deep hough transform for semantic line detection. In *ECCV*, pages 750–766, 2020.

[15] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2000. ISBN 0521623049.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[18] Lam Huynh, Phong Nguyen-Ha, Jiri Matas, Esa Rahtu, and Janne Heikkilä. Guiding monocular depth estimation using depth-attention volume. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 581–597, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58574-7.

[19] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[21] Jae-Han Lee, Minhyeok Heo, Kyung-Rae Kim, and Chang-Su Kim. Single-image depth estimation based on fourier domain analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 330–339, 2018.

[22] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3D detection from object keypoints for autonomous driving. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 644–660, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58580-8.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[25] Yancong Lin, Silvia L Pintea, and Jan C van Gemert. Deep hough-transform line priors. In *European Conference on Computer Vision*, pages 323–340. Springer, 2020.

[26] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, 2018.

[27] Yuxuan Liu, Yuan Yixuan, and Ming Liu. Ground-aware monocular 3D object detection for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):919–926, 2021.

[28] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, April 2017.

[29] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 311–327, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58601-0.

[30] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[31] Nikola Popovic, Danda Pani Paudel, Thomas Probst, Guolei Sun, and Luc Van Gool. Compositetasking: Understanding images by spatial composition of tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6870–6880, June 2021.

[32] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[33] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[35] Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popovic, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool. Task switching network for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8291–8300, October 2021.

[36] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[37] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5354–5362, 2017.

[38] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5684–5693, 2019.

[39] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[40] Ruo Zhang, Ping-Sing Tsai, J.E. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999. doi: 10.1109/34.784284.