Data-efficient learning of geometric structures from single-view images

Lin, Y.

DOI
10.4233/uuid:d0f89fb8-bb07-499d-bff4-29b074ed17ef

Publication date
2022

Document Version
Final published version

Citation (APA)
Lin, Y. (2022). *Data-efficient learning of geometric structures from single-view images*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:d0f89fb8-bb07-499d-bff4-29b074ed17ef

DATA-EFFICIENT LEARNING OF GEOMETRIC STRUCTURES FROM SINGLE-VIEW IMAGES

DATA-EFFICIENT LEARNING OF GEOMETRIC STRUCTURES FROM SINGLE-VIEW IMAGES

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 25 April 2022 at 15:00 o'clock

by

Yancong LIN

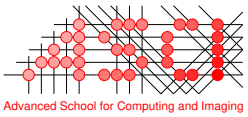
Master of Engineering in Electronic and Communication Engineering,
Tianjin University, China
Born in Qixia, Shandong, China

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. ir. M. J. T. Reinders,	Delft University of Technology, promotor
Dr. J. C. van Gemert,	Delft University of Technology, copromotor
Dr. S. L. Pintea,	Delft University of Technology, copromotor
Independent members:	
Prof. dr. D. M. Gavrila	Delft University of Technology
Prof. dr. K. J. Batenburg	Leiden University
Prof. dr. T. Gevers	University of Amsterdam
Prof. dr. R. C. Veltkamp	Utrecht University
Prof. dr. ir. R. L. Lagendijk	Delft University of Technology, reserve member

This research was funded by the China Scholarship Council.



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 434.

Printed by: ProefschriftMaken

Cover illustration: Joke Krul, Delft 2020 ©

Background photo: Marjolein van der Veldt

ISBN 978-94-6423-778-8

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

CONTENTS

1

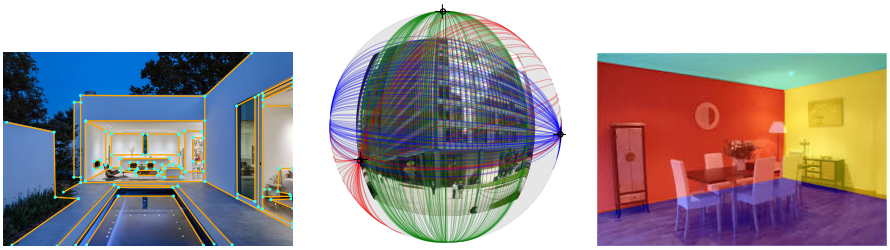
INTRODUCTION

Deriving geometric representations from images of the humanly constructed world is a long-standing task in computer vision [1, 2]. The main objective is to characterize the surroundings that we are interacting with into rich geometric and semantic structures, such as wireframes and planar surfaces, as shown in Figure 1.1. These structures are important cues that we can rely on for visual navigation and scene understanding. For instance, we can estimate the spatial layout inside a building by relying on a handful of patterns, e.g., floors, walls and ceilings. The ability to effortlessly infer these patterns is an important advantage of human beings over autonomous agents. A natural extension is how to impart this ability to autonomous agents such that they can automatically recognize, model and analyze geometric structures in the artificial world.

Humans can intuitively characterize artificial structures into a set of geometric primitives, including junctions, line segments, parametric curves and planes, each of which defines a collection of elements that share the same geometry or semantics, such as appearance, shape, orthogonality, parallelism and symmetry [5]. Consider an architectural drawing as an example where 3D geometry such depth, scale, distance, can be linked to geometric primitives such as line segments to denote the outline of a building, vanishing points to encode parallel lines, and intersecting planes to indicate orthogonality. These primitives provide compact geometric information of a visual scene, facilitating the understanding of the humanly constructed world.

Research on geometric primitives stems as early as from the 1970's where the goal is to recognise visual concepts from simple primitives [6]. Many techniques have been proposed to automatically detect geometric primitives from digital images since then. One well-known example is the Hough Transform [7] - a prevalent algorithm for straight line detection. Lines are low-level edge features by definition but are able to capture high-level semantics, empowering autonomous driving [8, 9] and architectural design [10]. Moreover, lines also encode crucial 3D geometry since 3D parallel lines intersect at a vanishing point when projected onto an image plane [11]. Detecting vanishing points allows us to infer 3D orientation of lines from single 2D images without relying on 3D supervision, thus enjoying great popularity in 3D scene understanding [12–14]. Another important property of

1



(a) Wireframes (b) Vanishing points (c) Planar surfaces

Figure 1.1: **Geometric primitives in the humanly constructed world.** (a) Wireframes describe the topology of line segments [2]. (b) Vanishing points (the intersections of colored lines) represent the orientation of 3D parallel lines [3]. (c) Orthogonal planes characterize the layout of an indoor scene [4]. The overall goal of this thesis is to extract these geometric primitives from a single image.

artificial objects is symmetry, including mirror, rotation and translation symmetries. Symmetries have been proven beneficial in numerous 3D reconstruction tasks, particularly in shape completion [15, 16]. Therefore, it is an essential task in computer vision to accurately and efficiently identify diverse geometric primitives.

Classic work on geometric primitive detection relies on well-designed, manually constructed features, where the geometric knowledge is explicitly incorporated into feature engineering [17]. These approaches often start with edges and then group edges into diverse primitives depending on the task. For instance, a conventional line detection approach first applies the Canny edge detector [18] followed by the Hough Transform to aggregate edge features into lines. Although there has been significant progress over decades [19, 20], classic approaches are unable to produce reliable predictions in challenging scenarios due to incorrect edge pixel identification [21]. Therefore, researchers have been looking for alternatives that are less dependent on manually designed features, where learning features from labeled data with deep neural networks is promising.

Research on deep neural networks has seen an explosive growth in computer vision, because deep networks improve accuracy on numerous tasks [22, 23]. The key of neural networks is to learn the optimal parameters that minimize the discrepancy between the model predictions and the target values. Given an image $\mathbf{x}$, a target value y, and a neural network f_Θ parameterized by Θ, we minimize the loss L with respect to Θ:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{n=1}^{N} L(f_\Theta(\mathbf{x}^n), y^n), \tag{1.1}$$

where N is the total number of training samples. The loss L measures the deviation between the prediction $f_\Theta(\mathbf{x}^n)$ and the target value y^n. Typically, a neural network contains millions of parameters and requires a tremendous amount of labeled samples to learn the optimal parameter Θ [24, 25]. However, labeling data can be notoriously expensive, making data annotation costs a crucial factor in training neural networks [22].

There are several options to mitigate the necessity of exhaustively labeled datasets, e.g., adding inductive prior knowledge [26, 27], leveraging massive unlabeled data [28, 29], etc. One representative of adding priors is classic feature engineering, which typically

does not require large datasets, because the geometric priors are explicitly encoded into feature extraction. Therefore, it would be beneficial to incorporate various geometric priors into neural networks to reduce the annotation effort. However, training neural networks relies on gradient descent, an algorithm that calculates the gradients of the loss L with respect to the parameter Θ [24]. Thus, differentiability is a prerequisite for training neural networks. In practice, it is often preferred to optimize all the parameters jointly in an end-to-end fashion. Consequently, the challenge is to implement diverse geometric priors as differentiable modules for end-to-end training. In this thesis, we explore the possibility of adding various priors into neural networks for data-efficient learning.

Leveraging large quantities of unlabeled images is another option to reduce the dependence on costly manual annotation, although it does demand matching computational resources. Semi-supervised learning is such a learning paradigm that combines a small fraction of labeled data and a large portion of unlabeled data. This approach has been adopted in numerous vision tasks, e.g., image classification [28, 29], semantic segmentation [30], etc. A simple yet effective technique in semi-supervised learning is pseudo-labeling [31], where a model is first trained on the small amount of labeled data only, and then used to generate predictions for the unlabeled data. The predictions with high confidence are considered as correct pseudo-labels which can be used to train a new model using both the ground truth labels and pseudo-labels. In addition to adding priors, we also study the usage of pseudo-labeling in leveraging unlabeled images to improve data efficiency.

One of the main goals of this thesis is to detect geometric primitives from single-view images with deep neural networks. Our interest is in reducing the dependency on large manually labeled datasets and in improving the overall performance in a small data regime, by incorporating geometric knowledge into learning. The challenge is to implement geometric priors as end-to-end modules such that we can optimize the whole system through gradient back-propagation. The geometric primitives we are interested in are straight lines, vanishing points, mirror planes and polygons, corresponding to the following five tasks: (1) wireframe parsing, (2) traffic lane detection, (3) vanishing point detection, (4) 3D mirror symmetry detection, and (5) polygonal shape detection.

We first give a brief introduction of geometric priors. Then we elaborate each task in an individual chapter. We conclude with our contributions of this thesis.

1.1. GEOMETRIC PRIORS

Prior knowledge refers to specialized expertise which can be useful in a task, apart from training data. A simple example is data augmentation in image classification [32]. The knowledge is that certain image transformations, such as rotations and flips, typically do not alter the class label of a given image. Therefore, it is common to generate multiple duplicates of an image by applying these transformations to increase the value of a single data sample during training. Another example is the widely used feature pyramid network [33], which handles scale variation in object detection as photographed objects may differ significantly in size. The basic idea is to aggregate feature representations at multiple scales such that smaller objects can also be correctly identified. This design is largely inspired by the Gaussian and Laplacian pyramids [34, 35] and the scale-invariant feature transform (SIFT) [36], indicating the significance of expert knowledge in both feature engineering and deep learning.

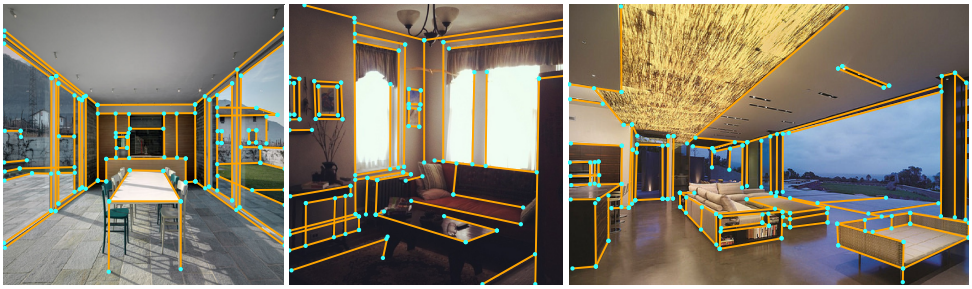


Figure 1.2: **Wireframe parsing.** Wireframes provide reach geometric information for scene understanding [2].

Adding expert knowledge to deep learning restricts the parameter space, and thus has a regularizing effect, offering potential in parameter reduction, data efficiency and performance improvements. The best-known example is arguably the translation equivariance of convolutional networks which is a particularly successful model for image object detection. The prior knowledge is that an object may appear anywhere inside an image, and thus it is important to share filters across the entire space [32]. This not only enables neural networks to recognize patterns regardless of their positions, but also significantly reduces parameters. The prevalence of convolutional networks has verified the importance of incorporating prior knowledge into deep learning.

There are a variety of priors in vision, ranging from visual appearance to camera geometry. Appearance-based priors describe *what an object looks like* [37], while geometry-based priors illustrate *how an image is captured* [38]. Visual appearance is mostly defined by pixel colors (RGB values), which are jointly determined by light sources, surface normals, camera views and materials. Once a variable changes, the recorded color varies accordingly. This knowledge can be encoded in a physical reflection model [37, 39, 40], without learning from data. Camera geometry defines the conversion from a ray to a pixel. Common knowledge, such as the pinhole camera and multi-view geometry, has been an indispensable prior in 3D vision [38]. A renowned example is the Epipolar geometry [41], which offers a strong prior to eliminate false matches in image matching. Although being domain-specific, geometric priors contribute to data efficiency as they restrict the solution space.

In the following, we briefly introduce the geometric priors as linked to the chapters.

1.1.1. DEEP HOUGH TRANSFORM LINE PRIORS

Line segments are a common geometric primitive in a humanly constructed world as shown in Figure 1.2. They are also referred to as wireframes, together with junctions [2]. Wireframes are useful in many aspects, such as architectural design [10], room layout estimation [13], and 3D reconstruction [14]. Recent work on wireframe detection leverages the capability of deep neural networks in learning from large annotated datasets and outperforms classic approaches by a large margin on the Wireframe benchmark [2]. However, the learning-based models show a significant performance decrease in certain scenarios where training data is scarce [42]. This brings us to the question: *"How to improve the overall performance of neural networks in a small-data regime?"*

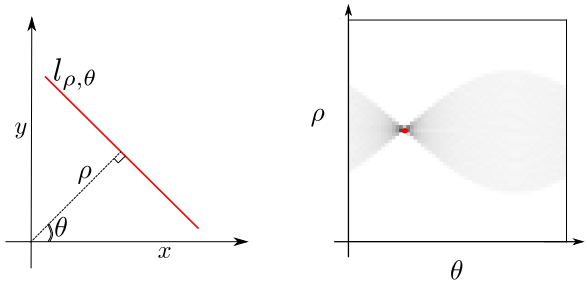


Figure 1.3: **Hough Transform.** The left figure shows a line l in the Euclidean space (x, y) with its offset-angle (ρ,θ) parameterization, where ρ is the offset from the origin and θ represents the orientation of the line. The right figure illustrates the Hough transform of the line. A straight line is equivalent to a local maximum (the red dot) in the Hough space.

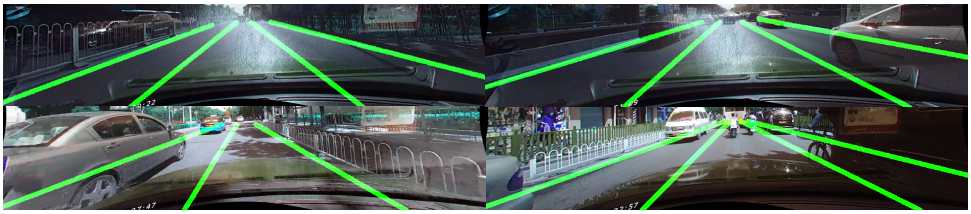


Figure 1.4: **Traffic lane detection.** The ground truth is annotated in green. Illumination variation and traffic flows are challenging for lane detection.

To answer this question, we consider the Hough Transform [7] as a global geometric prior of lines and explore the possibility of adding this prior into neural networks, because the knowledge of lines needs not to be learned from massive data. The inductive knowledge is that points on a straight line share the same angle (θ) and offset (ρ) in the polar coordinate system. Figure 1.3 shows an example of the Hough Transform, where a straight line is parameterized as a single bin (ρ, θ) in the Hough space, thus turning line detection into spotting individual bins. We offer a principled way to inject the Hough Transform line priors into neural networks for data-efficient wireframe parsing in chapter 2.

1.1.2. SEMI-SUPERVISED LANE DETECTION WITH DEEP HOUGH TRANSFORM

Autonomous driving [9] relies on precise lane detection, which is inherently a challenging task due to illumination variations and traffic flows, as shown in Figure 1.4. Moreover, popular datasets only capture a fraction of driving scenarios in real-world environments [43, 44], limiting the generalization ability of current approaches in other unseen driving scenarios. Therefore, leveraging additional realistic unlabelled data will be beneficial. This leads to the question: *"Can we design a semi-supervised model to exploit vast amounts of unlabeled data?"*

To this goal, we combine a pseudo-labeling strategy and the Hough Transform line priors for data-efficient lane detection in chapter 3.

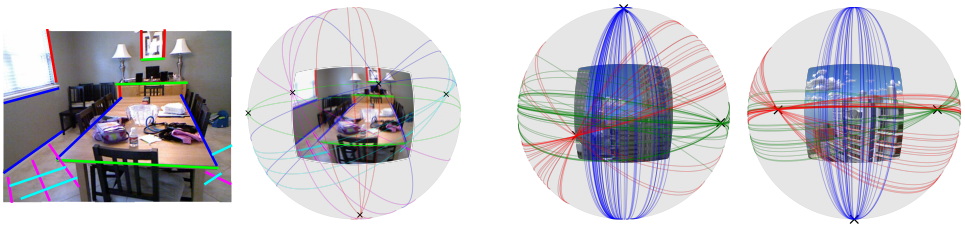


Figure 1.5: **Vanishing point detection.** The two pictures on the left display the line segments and associated vanishing points (in color) on the image plane and on the Gaussian sphere, respectively [47]. The two examples on the right display three orthogonal vanishing points in the Manhattan world [14].

1.1.3. GEOMETRIC PRIORS FOR DEEP VANISHING POINT DETECTION

One interesting observation in daily life is that parallel lines may intersect at a distance, such as railways, highways and riverbanks. Parallel lines never cross in 3D world, but they overlap at the same point in 2D images. The intersecting point is termed the vanishing point, which encodes the orientation of 3D parallel lines. Vanishing point enjoys great success in vision as it allows us to infer 3D geometry from a single image without any 3D supervision. Figure 1.5 shows examples of vanishing point detection from single images.

Conventional approaches on vanishing point detection work with line segments, as vanishing points are inherently intersections of lines. Therefore the line segment detector has a significant impact on the overall performance. Recently, deep neural networks have also been applied to the vanishing point detection task and show strong performance on large-scale datasets [45, 46]. The main advantage of these models is their ability to learn useful context information directly from images without purely relying on line segments. While showing substantial improvement over conventional baselines, the data-driven learning models are typically trained on large datasets with a vast number of parameters [45]. Moreover, the majority of works enforce the Manhattan world assumption which states that there are three orthogonal vanishing points [3]. However, this assumption is no longer applicable in the real world due to the presence of multiple non-orthogonal vanishing points. Those findings motivates us to the question: *"Can we find a prior independent of both large quantities of training samples and the Manhattan world assumption?"*

We exploit the Gaussian sphere mapping [11], a prior for vanishing points which maps lines from the unbounded image plane to a bounded hemisphere, as shown in Figure 1.6. Detecting vanishing points is equivalent to localizing spherical points in the alternative space, enabling neural networks to detect vanishing points outside the image view. We validate the added values of the Gaussian sphere mapping in neural networks for data efficiency, detecting multiple non-orthogonal vanishing points and transferability to unseen datasets in chapter 4.

1.1.4. DATA-EFFICIENT LEARNING FOR 3D MIRROR SYMMETRY DETECTION

Mirror symmetry has been proven effective in 3D shape completion [50, 51] and single-view 3D reconstruction [52–54], as it allows us to hallucinate the global structure from

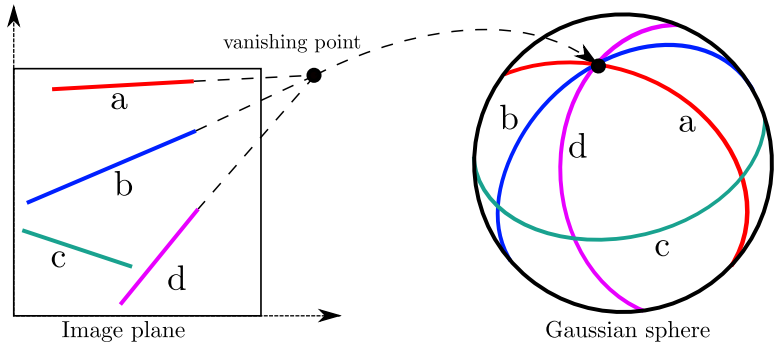


Figure 1.6: **Gaussian sphere mapping for vanishing point detection.** (a) Multiple lines intersect at the vanishing point on the image plane. (b) We show the projection of lines and vanishing point on the Gaussian sphere, where lines become great circles and vanishing point turns into a spherical point.

Figure 1.7: **Mirror symmetry in artificial objects.** We show the symmetric axis (the projection of the 3D mirror plane on the image plane) in green. Mirror symmetry is useful for inferring the the global structure from only partial observations.

only partial observations. While it is common to identify 3D symmetries from RGB-D images [16], acquiring depth is often challenging for average users due to the lack of depth cameras. We focus on detecting 3D mirror symmetry from single-view perspective images, thus removing the necessity of depth.

A common strategy to detect 3D mirror symmetry is correspondence matching followed by camera geometry reasoning [48, 49]. However, this approach does not generalize well to textureless objects and smooth surfaces due to lacking correspondences. Instead of local feature matching, the work in [55] proposes to learn semantic features directly from large amounts of data. The combination of feature learning and geometry reasoning has lead to top performance on public datasets [56, 57]. However, the demand of massive data remains a concern for real-world application. This prompts us to the question: *"Can we mitigate the dependency on large datasets by adding priors?"*

The prior we are interested in is the 3D mirror geometry, as illustrated in Figure 1.8, which provides an illustration of how to compute the symmetric correspondence of a point on the other side of its mirror plane in Euclidean space. This knowledge is particularly ben-

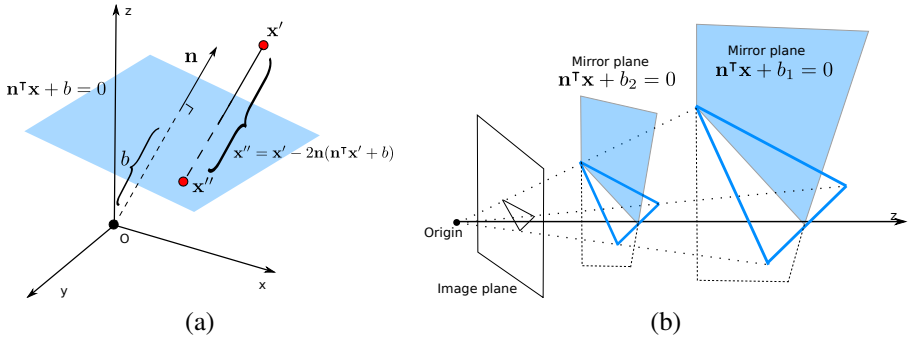


Figure 1.8: **3D mirror geometry and scale ambiguity.** (a) $\mathbf{x}'$ and $\mathbf{x}''$ are symmetric with respect to the plane $\mathbf{n}^\intercal\mathbf{x}+b=0$, where $\mathbf{n}$ and b are the normal and offset of the given plane. The mirror $\mathbf{x}''$ can be explicitly defined by $\mathbf{x}''=\mathbf{x}'-2\mathbf{n}(\mathbf{n}^\intercal\mathbf{x}'+b)$ [48, 49]. (b) Although differing in scale, the two objects (in blue) have exactly the same projections on the image plane. Thus, it is impossible to infer the actual size of an object (or the offset of the mirror plane), hence, there is scale ambiguity in detecting the mirror plane from 2D images.

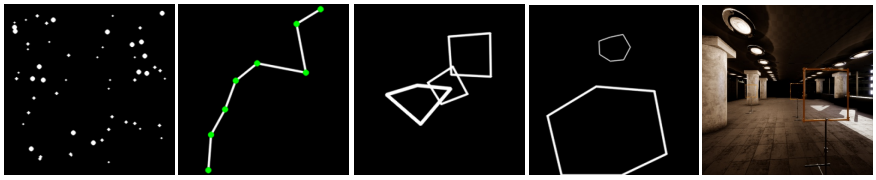


Figure 1.9: **Polygons as collections of points.** We consider polygons as collections of points, and study transformers in predicting point collections on four toy datasets, namely points, lines, gates and polygons, as well as a synthetic gate dataset.

eficial when an object is textureless since we are unable to match correspondences purely relying on appearance. Another knowledge is the scale ambiguity in detecting 3D mirror planes from 2D images, as we are not able to decide the absolute scale of an object or the offset the mirror plane, as indicated in Figure 1.8. Therefore, the only variable we are interested in is the normal direction of a mirror plane. We incorporate this knowledge into an end-to-end learning framework and experimentally demonstrate the impact of adding 3D mirror geometry in improving data efficiency, as illustrated in chapter 5.

1.1.5. INVESTIGATING TRANSFORMERS IN THE DECOMPOSITION OF POLYGONAL SHAPES AS POINT COLLECTIONS

Polygons are another common geometric primitive in a humanly constructed world, which consists of a chain of line segments or a collection of ordered vertices. Learning polygonal representations from artificial objects has the potential to assist numerous high-level applications, especially in aerial image processing. One popular example is to extract vectorized building outlines from high-resolution satellite images [58, 59].

Polygons have a varying cardinality (i.e., the number of vertices). For instance, a triangle has 3 vertices while a diamond has 4. Another important aspect is that the order of vertices matters, as the same set of vertices can form differently shaped polygons. However,

canonical convolutional neural networks lack the ability to incorporate these priors by design [60]. In comparison, the widely used auto-regressive transformers [61, 62] in machine translation and speech recognition [63, 64] are able to handle tasks with unknown cardinality and predefined order. Moreover, transformers have also been successfully applied to computer vision, e.g., parallel transformers for object detection where the output is a set of bounding boxes [65, 66]. The popularity of transformers prompts us to the question: *"Can we leverage transformers for detecting polygonal shapes?"*

To answer this question, we treat polygons as *collections* of points, such as sequences and sets, and study the difference between auto-regressive and parallel transformers on four toy datasets, including individual points, ordered lines, gates and polygons, as shown in Figure 1.9. We present our observations and conclusions in chapter 6.

1.2. CONTRIBUTION

The main contribution of this thesis is on how to incorporate classical geometric visual priors in deep neural networks. We validate the added value of geometric priors in data-efficient learning on five tasks: (1) wireframe parsing, (2) lane detection, (3) vanishing point detection, (4) 3D mirror symmetry detection and (5) polygonal shape detection.

Chapter 2 proposes to incorporate the Hough Transform into neural networks while facilitating end-to-end learning [42]. The proposed model contains a Hough Transform module which maps pixel-wise features into the Hough space where lines can be identified as local maxima, and an inverse Hough Transform module which maps individual bins back to pixels. Filtering in the Hough space allows us to detect lines more efficiently since the Hough Transform is able to aggregate information globally from all pixels on a line. In comparison, convolutional networks typically require a stack of layers to capture the entire line because the receptive field is only a local patch for each layer. Numerous experiments on two benchmarks show the added value of the Hough Transform line priors towards both parameter reduction and data efficiency.

Chapter 3 focuses on the usage of the Hough Transform line priors in learning from unlabeled data for lane detection. The main contribution is a novel loss function, which exploits the geometric representations of lanes in a semi-supervised manner. Our observation is that the presence of lanes leads to Hough bins with maximal votes. The intensity of Hough bins indicates the probability of a lane being present, while the position represents the layout (angle and offset) of a lane on the image plane, as shown in Figure 1.3. Having a large maximum indicates that pixels along that line direction are well aligned, thus falling in the same bin. Maximizing the log-probability of these Hough bins requires no human supervision, enabling neural networks to detect lanes from unlabelled images. Extensive experiments on two public datasets [43, 44] demonstrate the ability of the proposed loss function in learning from unlabeled data, especially when annotations are scarce, e.g., 1% of the whole dataset.

Chapter 4 explores Gaussian sphere mapping - a geometric prior for vanishing point detection to enrich neural networks when used to detect multiple non-orthogonal vanishing points without large datasets. The main idea is to project line segments from the image plane to a unit hemisphere, where each line segment becomes a great circle and the intersection of multiple great circles represents a vanishing point, as illustrated in Figure 1.6. Our Gaussian sphere mapping is implemented as an end-to-end module, making it possi-

ble to learn powerful feature representations directly from images without the need for line segment detectors. Moreover, we can apply common clustering techniques to find multiple non-orthogonal vanishing points on the unit hemisphere, making our model applicable to both Manhattan and non-Manhattan scenarios. Extensive experiments demonstrate the competitiveness of our model on multiple datasets, especially when training with a fraction of labeled data (e.g., 500 images only). More importantly, we improve over previous state-of-the-art considerably in the challenging non-Manhattan scenario, as well as cross-dataset tests with domain shift.

Chapter 5 studies detecting 3D mirror symmetry from single-view perspective images. We introduce 3D mirror geometry as a prior for calculating mirrored correspondences. This prior allows us to compute the correlation between a pixel and its mirrors at various depth, which indicates the extent to which a pixel resembles its mirrors with regard to the given plane. Another geometric prior is the scale ambiguity as we are unable to determine the offset of the mirror plane from a single-view image, as shown in Figure 1.8. Therefore, we are only interested in identifying the normal direction of the mirror plane. Given that the normal direction of all possible planes lies on a unit sphere, we design multi-stage spherical convolutions to pinpoint the exact mirror plane in a coarse-to-fine manner. Our model not only improves the data efficiency and overall performance but also reduces the inference time by approximately a factor of 20 over the top-performing model [55].

Chapter 6 shifts the focus to polygons - a structure composed of a *collection* of points. Particularly, we investigate auto-regressive and parallel transformers on detecting four polygonal shapes: points, lines, gates and polygons, as shown in Figure 1.9. We provide a full picture of the advantages and disadvantages of both types of transformers, and show empirically the impact of the order in a collection of points. More importantly, we show empirically that auto-regressive transformers are a viable solution to detect a set of polygons when the elements inside each polygon are an *ordered* set of points.

Chapter 7 summarizes this thesis and presents possible extensions for future work.

REFERENCES

[1] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, *State of the art in surface reconstruction from point clouds,* in *Eurographics 2014-State of the Art Reports*, Vol. 1 (2014) pp. 161–185.

[2] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, *Learning to parse wireframes in images of man-made environments,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 626–635.

[3] P. Denis, J. H. Elder, and F. J. Estrada, *Efficient edge-based methods for estimating manhattan frames in urban imagery,* in *European conference on computer vision* (Springer, 2008) pp. 197–210.

[4] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, *3d room layout estimation from a single rgb image,* IEEE Transactions on Multimedia **22**, 3014 (2020).

[5] Y. Zhou, S. Liu, and Y. Ma, *Learning to detect 3d reflection symmetry for single-view reconstruction,* arXiv preprint arXiv:2006.10042 (2020).

[6] M. B. Clowes, *On seeing things,* Artificial intelligence **2**, 79 (1971).

[7] R. O. Duda and P. E. Hart, *Use of the hough transformation to detect lines and curves in pictures,* Communications of the ACM **15**, 11 (1972).

[8] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, *Recent progress in road and lane detection: a survey,* Machine vision and applications **25**, 727 (2014).

[9] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, *Learning lightweight lane detection cnns by self attention distillation,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 1013–1021.

[10] Y. Zhou, H. Qi, and Y. Ma, *End-to-end wireframe parsing,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 962–971.

[11] S. T. Barnard, *Interpreting perspective images,* Artificial intelligence **21**, 435 (1983).

[12] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon, *Vpgnet: Vanishing point guided network for lane and road marking detection and recognition,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 1947–1955.

[13] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, *Layoutnet: Reconstructing the 3d room layout from a single rgb image,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 2051–2059.

[14] Y. Zhou, H. Qi, Y. Zhai, Q. Sun, Z. Chen, L.-Y. Wei, and Y. Ma, *Learning to reconstruct 3d manhattan wireframes from a single image,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 7698–7707.

[15] L. Gao, L.-X. Zhang, H.-Y. Meng, Y.-H. Ren, Y.-K. Lai, and L. Kobbelt, *Prs-net: Planar reflective symmetry detection net for 3d models,* arXiv preprint arXiv:1910.06511 (2019).

[16] Y. Shi, J. Huang, H. Zhang, X. Xu, S. Rusinkiewicz, and K. Xu, *Symmetrynet: learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images,* ACM Transactions on Graphics (TOG) **39**, 1 (2020).

[17] D. G. Lowe, *Object recognition from local scale-invariant features,* in *Proceedings of the seventh IEEE international conference on computer vision*, Vol. 2 (Ieee, 1999) pp. 1150–1157.

[18] J. Canny, *A computational approach to edge detection,* IEEE Transactions on pattern analysis and machine intelligence , 679 (1986).

[19] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *Lsd: A fast line segment detector with a false detection control,* IEEE transactions on pattern analysis and machine intelligence **32**, 722 (2008).

[20] L. Quan and R. Mohr, *Determining perspective structures using hierarchical hough transform,* Pattern Recognition Letters **9**, 279 (1989).

[21] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, *Learning attraction field representation for robust line segment detection,* in *The IEEE Conference on Computer Vision and Pattern Recognition* (2019).

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database,* in *2009 IEEE conference on computer vision and pattern recognition* (Ieee, 2009) pp. 248–255.

[23] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[24] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning,* nature **521**, 436 (2015).

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,* Advances in neural information processing systems **25**, 1097 (2012).

[26] O. S. Kayhan and J. C. v. Gemert, *On translation invariance in cnns: Convolutional layers can exploit absolute spatial location,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 14274–14285.

[27] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders, *Structured receptive fields in cnns,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 2610–2619.

[28] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, *Fixmatch: Simplifying semi-supervised learning with consistency and confidence,* arXiv preprint arXiv:2001.07685 (2020).

[29] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations,* in *International conference on machine learning* (PMLR, 2020) pp. 1597–1607.

[30] S. Mittal, M. Tatarchenko, and T. Brox, *Semi-supervised semantic segmentation with high-and low-level consistency,* IEEE transactions on pattern analysis and machine intelligence (2019).

[31] A. Chaudhary, *Semi-supervised learning in computer vision,* (2020), https://amitness.com/2020/07/semi-supervised-learning/.

[32] M. Rath and A. P. Condurache, *Boosting deep neural networks with geometrical prior knowledge: A survey,* arXiv preprint arXiv:2006.16867 (2020).

[33] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017) pp. 2117–2125.

[34] T. Lindeberg, *Edge detection and ridge detection with automatic scale selection,* International journal of computer vision **30**, 117 (1998).

[35] P. J. Burt and E. H. Adelson, *The laplacian pyramid as a compact image code,* in *Readings in computer vision* (Elsevier, 1987) pp. 671–679.

[36] D. G. Lowe, *Distinctive image features from scale-invariant keypoints,* International journal of computer vision **60**, 91 (2004).

[37] B. T. Phong, *Illumination for computer generated pictures,* Communications of the ACM **18**, 311 (1975).

[38] A. M. Andrew, *Multiple view geometry in computer vision,* Kybernetes (2001).

[39] S. A. Shafer, *Using color to separate reflection components,* Color Research & Application **10**, 210 (1985).

[40] A. Lengyel, S. Garg, M. Milford, and J. C. van Gemert, *Zero-shot domain adaptation with a physics prior,* arXiv preprint arXiv:2108.05137 (2021).

[41] R. I. Hartley, *In defense of the eight-point algorithm,* IEEE Transactions on pattern analysis and machine intelligence **19**, 580 (1997).

[42] Y. Lin, S. L. Pintea, and J. C. van Gemert, *Deep hough-transform line priors, European Conference on Computer Vision*, (2020).

[43] *TuSimple Lane Detection Challenge*.

[44] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, *Spatial as deep: Spatial cnn for traffic scene understanding,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32 (2018).

[45] Y. Zhou, H. Qi, J. Huang, and Y. Ma, *Neurvps: Neural vanishing point scanning via conic convolution,* arXiv preprint arXiv:1910.06316 (2019).

[46] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, *Scannet: Richly-annotated 3d reconstructions of indoor scenes,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 5828–5839.

[47] F. Kluger, E. Brachmann, H. Ackermann, C. Rother, M. Y. Yang, and B. Rosenhahn, *Consac: Robust multi-model fitting by conditional sample consensus,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 4634–4643.

[48] D. Cailliere, F. Denis, D. Pele, and A. Baskurt, *3d mirror symmetry detection using hough transform,* in *2008 15th IEEE International Conference on Image Processing* (IEEE, 2008) pp. 1772–1775.

[49] K. Köser, C. Zach, and M. Pollefeys, *Dense 3d reconstruction of symmetric scenes from a single image,* in *Joint Pattern Recognition Symposium* (Springer, 2011) pp. 266–275.

[50] L. Gao, L. X. Zhang, H. Y. Meng, Y. H. Ren, Y. K. Lai, and L. Kobbelt, *Prs-net: Planar reflective symmetry detection net for 3d models,* IEEE Transactions on Visualization and Computer Graphics , 1 (2020).

[51] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, *Morphing and sampling network for dense point cloud completion,* in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34 (2020) pp. 11596–11603.

[52] Y. Yao, N. Schertler, E. Rosales, H. Rhodin, L. Sigal, and A. Sheffer, *Front2back: Single view 3d shape reconstruction via front to back prediction,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 531–540.

[53] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, *Disn: Deep implicit surface network for high-quality single-view 3d reconstruction,* arXiv preprint arXiv:1905.10711 (2019).

[54] S. Wu, C. Rupprecht, and A. Vedaldi, *Unsupervised learning of probably symmetric deformable 3d objects from images in the wild,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 1–10.

[55] Y. Zhou, S. Liu, and Y. Ma, *Nerd: Neural 3d reflection symmetry detector,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 15940–15949.

[56] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, *Shapenet: An information-rich 3d model repository,* arXiv preprint arXiv:1512.03012 (2015).

[57] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, *Pix3d: Dataset and methods for single-image 3d shape modeling,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 2974–2983.

[58] N. Girard, D. Smirnov, J. Solomon, and Y. Tarabalka, *Polygonal building extraction by frame field learning,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 5891–5900.

[59] Z. Li, J. D. Wegner, and A. Lucchi, *Topological map extraction from overhead images,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 1715–1724.

[60] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, *Set transformer: A framework for attention-based permutation-invariant neural networks,* in *International Conference on Machine Learning* (PMLR, 2019) pp. 3744–3753.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need,* in *Advances in neural information processing systems* (2017) pp. 5998–6008.

[62] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate,* arXiv preprint arXiv:1409.0473 (2014).

[63] J. Guo, X. Tan, D. He, T. Qin, L. Xu, and T.-Y. Liu, *Non-autoregressive neural machine translation with enhanced decoder input,* in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33 (2019) pp. 3723–3730.

[64] V. Ganapathi-Subramanian, O. Diamanti, S. Pirk, C. Tang, M. Niessner, and L. Guibas, *Parsing geometry using structure-aware shape templates,* in *2018 International Conference on 3D Vision (3DV)* (IEEE, 2018) pp. 672–681.

[65] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers,* in *European Conference on Computer Vision* (Springer, 2020) pp. 213–229.

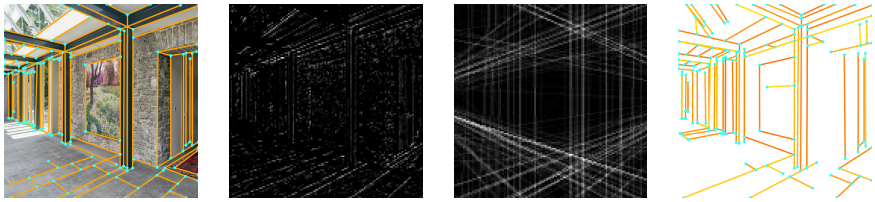
[66] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, *Deformable detr: Deformable transformers for end-to-end object detection,* arXiv preprint arXiv:2010.04159 (2020).

2

DEEP HOUGH-TRANSFORM LINE PRIORS

Classical work on line segment detection is knowledge-based; it uses carefully designed geometric priors using either image gradients, pixel groupings, or Hough transform variants. Instead, current deep learning methods do away with all prior knowledge and replace priors by training deep networks on large manually annotated datasets. Here, we reduce the dependency on labeled data by building on the classic knowledge-based priors while using deep networks to learn features. We add line priors through a trainable Hough transform block into a deep network. Hough transform provides the prior knowledge about global line parameterizations, while the convolutional layers can learn the local gradient-like line features. On the Wireframe (ShanghaiTech) and York Urban datasets we show that adding prior knowledge improves data efficiency as line priors no longer need to be learned from data.

This chapter is published as:
Yancong Lin, Silvia-Laura Pintea, and Jan van Gemert. *Deep Hough-Transform Line Priors*. European Conference on Computer Vision (ECCV), 2020. arXiv: https://arxiv.org/abs/2007.09493.



Ground truth | Learned local features | Added line priors | Line predictions

Figure 2.1: We add prior knowledge to deep networks for data efficient line detection. We learn local deep features, which are combined with a global inductive line priors, using the Hough transform. Adding prior knowledge saves valuable training data.

2.1. INTRODUCTION

Line segment detection is a classic Computer Vision task, with applications such as road-line detection for autonomous driving [1–4], wireframe detection for design in architecture [5–7], horizon line detection for assisted flying [8–10], image vectorization [11, 12]. Such problems are currently solved by state-of-the-art line detection methods [5, 6, 13] by relying on deep learning models powered by huge, annotated, datasets.

Training deep networks demands large datasets [14, 15], which are expensive to annotate. The amount of needed training data can be significantly reduced by adding prior knowledge to deep networks [16–18]. Priors encode inductive solution biases: e.g. for image classification, objects can appear at any location and size in the input image. The convolution operation adds a translation-equivariance prior [18, 19], and multi-scale filters add a scale-invariance prior [20, 21]. Such priors offer a strong reduction in the amount of required data: built-in knowledge no longer has to be learned from data. Here, we study straight line detection which allows us to exploit the line equation.

In this work we add geometric line priors into deep networks for improved data efficiency by relying on the Hough transform. The Hough transform has a long and successful history for line detection [22–24]. It parameterizes lines in terms of two geometric terms: an offset and an angle, describing the line equation in polar coordinates. This gives a global representation for every line in the image. As shown in figure 2.1, global information is essential to correctly locate lines, when the initial detections are noisy. In this work we do not exclusively rely on prior knowledge as in the classical approach [25–28] nor do we learn everything in deep architectures [5, 6, 13]. Instead, we take the best of both: we combine learned global shape priors with local learned appearance.

This paper makes the following contributions: (1) we add global geometric line priors through Hough transform into deep networks; (2) we improve data efficiency of deep line detection models; (3) we propose a well-founded manner of adding the Hough transform into an end-to-end trainable deep network, with convolutions performed in the Hough domain over the space of all possible image-line parameterizations; (4) we experimentally show improved data efficiency and a reduction in parameters on two popular line segment detection datasets, Wireframe (ShanghaiTech) [5] and York Urban [29].

2.2. RELATED WORK

Image Gradients. Lines are edges, therefore substantial work has focused on line segment detection using local image gradients followed by pixel grouping strategies such a region growing [27, 28], connected components [25], probabilistic graphical models [26]. Instead of knowledge-based approach for detecting local line features, we use deep networks to learn local appearance-based features, which we combine with a global Hough transform prior.

Hough transform. The Hough transform is the most popular algorithm for image line detection where the offset-angle line parameterization was first used in 1972 [22]. Given its simplicity and effectiveness, subsequent line-detection work followed this approach [23, 30, 31], by focusing on analyzing peaks in Hough space. To overcome the sensitivity to noise, previous work proposed statistical analysis of Hough space [32], and segment-set selection based on hypothesis testing [33]. Similarly, a probabilistic Hough transform for line detection, followed by Markov Chain modelling of candidate lines is proposed in [34], while [24] creates a progressive probabilistic Hough transform, which is both faster and more robust to noise. An extension of Hough transform with edge orientation is used in [35]. Though less common, the slope-intercept parameterization of Hough transform for detecting lines is considered in [36]. In [37] Hough transform is used for detecting page orientation for character recognition. In our work, we do not use hand-designed features, but exploit the line prior knowledge given by the Hough transform when included into a deep learning model, allowing it to behave as a global line-pooling unit.

Deep learning for line detection The deep network in [5] uses two heads: one for junction prediction and one for line detection. This is extended in [6], by a line-proposal subnetwork. A segmentation-network backbone combined with an attraction field map, where pixels vote for their closest line is used in [13]. Similarly, attraction field maps are also used in [38] for generating line proposals in a deep architecture. Applications of line prediction using a deep network include aircraft detection [39], and power-line detection [40]. Moving from 2D to 3D, [7] predicts 3D wireframes from a single image by relying on the assumption that image scenes have an underlying Cartesian grid. Another variation of the wireframe-prediction task is proposed in [13] which creates a fisheye-distorted wireframe dataset and proposes a method to rectify it. A graph formulation [41] can learn the association between end-points. The need for geometric priors for horizon line detection is investigated in [42], concluding that CNNs (Convolutional Neural Networks) can learn without explicit geometric information. However, as the availability of labeled data is a bottleneck, we argue that prior geometric information offers improved data efficiency.

Hough transform hybrids Using a vote accumulator for detecting image structure is used in [43] for curve detection. Deep Hough voting schemes are considered in [44] for detecting object centroids on 3D point clouds, and for finding image correspondences [45]. In our work, we also propose a Hough-inspired block that accumulates line votes from input featuremaps. The Radon transform is a continuous version of the Hough transform [46–48]. Inverting the Radon transform back to the image domain is considered in [49, 50]. In [50] an exact inversion from partial data is used, while [49] relies on a deep network for the inversion, however the backprojection details are missing. Related to Radon transform, the ridgelet transform [51] maps points to lines, and the Funnel transform detects

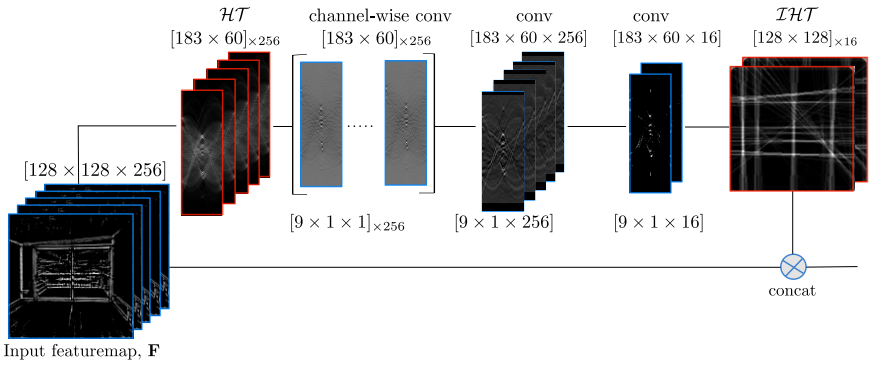


Figure 2.2: **HT-IHT block:** The input featuremap, **F**, coming from the previous convolutional layer, learns local edge information, and is combined on a residual branch with line candidates, detected in global Hough space. The input featuremap of $128 \times 128 \times 256$ is transformed channel-wise to the Hough domain through the $\mathcal{HT}$ layer into multiple 183×60 maps. The result is filtered with $1D$ channel-wise convolutions. Two subsequent $1D$ convolutions are added for merging and reducing the channels. The output is converted back to the image domain by the $\mathcal{IHT}$ layer. The two branches are concatenated together. Convolutional layers are shown in blue, and in red the $\mathcal{HT}$ and $\mathcal{IHT}$ layers. Our proposed HT-IHT block can be used in any architecture.

lines by accumulating votes using the slope-intercept line representation [52]. Similar to these works, we take inspiration from the Radon transform and its inversion in defining our Hough transform block.

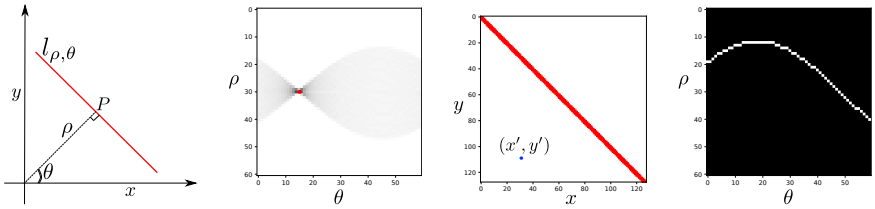
2.3. HOUGH TRANSFORM BLOCK FOR GLOBAL LINE PRIORS

Typically, the Hough transform parameterizes lines in polar coordinates as an offset ρ and an angle, θ. These two parameters are discretized in bins. Each pixel in the image votes in all line-parameter bins to which that pixel can belong. The binned parameter space is denoted the Hough space and its local extrema correspond to lines in the image. For details, see figure 2.3.(a,b) and [22].

We present a Hough transform and inverse Hough transform (HT-IHT block) to combine local learned image features with global line priors. We allow the network to combine information by defining the Hough transform on a separate residual branch. The $\mathcal{HT}$ layer inside the HT-IHT block maps input featuremaps to the Hough domain. This is followed by a set of local convolutions in the Hough domain which are equivalent to global operations in the image domain. The result is then inverted back to the image domain using the $\mathcal{IHT}$ layer, and it is subsequently concatenated with the convolutional branch. Figure 2.2 depicts our proposed HT-IHT block, which can be used in any architecture. To train the HT-IHT block end-to-end, we must specify its forward and backward definitions.

2.3.1. $\mathcal{HT}$: FROM IMAGE DOMAIN TO HOUGH DOMAIN

Given an image line $l_{\rho,\theta}$ in polar coordinates, with an offset ρ and angle θ, as depicted in figure 2.3.(a), for the point $P = (P_x, P_y)$ located at the intersection of the line with its normal, it holds that: $(P_x, P_y) = (\rho\cos\theta, \rho\sin\theta)$. A point along this line $(x(i), y(i))$ is given



(a) Input line (b) Line $\mathcal{HT}$ (c) Line $\mathcal{IHT}$ (d) Mask $\mathbf{B}(x', y')$

Figure 2.3: (a) A line together with its (ρ,θ) parameterization. (b) The Hough transform ($\mathcal{HT}$) of the line. (c) The inverse Hough transform ($\mathcal{IHT}$) of the Hough map. (d) The binary mask $\mathbf{B}$, mapping the pixel location (x', y') highlighted in blue in (c) to its corresponding set of bins in the Hough domain.

by:

$$(x(i), y(i)) = (\rho\cos\theta - i\sin\theta, \rho\sin\theta + i\cos\theta), \tag{2.1}$$

where $x(\cdot)$ and $y(\cdot)$ define the infinite set of points along the line as functions of the index of the current point, i, where $i \in \mathbb{R}$ can take both positive and negative values. Since images are discrete, here $(x(i), y(i))$ refers to the pixel indexed by i along an image direction.

The traditional Hough transform [22, 24] uses binary input where featuremaps are real valued. Instead of binarizing the featuremaps, we define the Hough transform similar to the Radon transform [46]. Therefore for a certain (ρ,θ) bin, our Hough transform accumulates the featuremap activations $\mathbf{F}$ of the corresponding pixels residing on that image direction:

$$\mathcal{HT}(\rho,\theta) = \sum_i \mathbf{F}_{\rho,\theta}(x(i), y(i)), \tag{2.2}$$

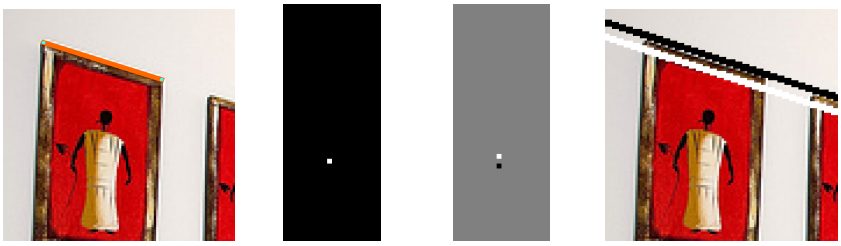
where the relation between the pixel $(x(i), y(i))$ and bin (ρ,θ) is given in equation (2.1), and $\mathbf{F}_{\rho,\theta}(x(i), y(i))$ is the featuremap value of the pixel indexed by i along the (ρ,θ) line in the image. The $\mathcal{HT}$ is computed channel-wise, but for simplicity, we ignore the channel dimension here. Figure 2.3.(b) shows the Hough transform map for the input line in figure 2.3.(a), where we highlight in red the bin corresponding to the line.

Note that in equation (2.2), there is a correspondence between the pixel $(x(i), y(i))$ and the bin (ρ,θ). We store this correspondence in a binary matrix, so we do not need to recompute it. For each featuremap pixel, we remember in which $\mathcal{HT}$ bins it votes, and generate a binary mask $\mathbf{B}$ of size: $[W, H, N_\rho, N_\theta]$ where $[W, H]$ is the size of the input featuremap $\mathbf{F}$, and $[N_\rho, N_\theta]$ is the size of the $\mathcal{HT}$ map. Thus, in practice when performing the Hough transform, we multiply the input feature map $\mathbf{F}$ with $\mathbf{B}$, channel-wise:

$$\mathcal{HT} = \mathbf{FB}. \tag{2.3}$$

Additionally, we normalize the $\mathcal{HT}$ by the width of the input featuremap.

We transform to the Hough domain for each featuremap channel by looping over all input pixels, $\mathbf{F}$, rather than only the pixels along a certain line, and we consider a range of discrete line parameters, (ρ,θ) where the pixels can vote. The (ρ,θ) pair is mapped into Hough bins by uniformly sampling 60 angles in the range $[0,\pi]$ and 183 offsets in the range $[0, d]$, where d is the image diagonal, and the computed offsets from θ are assigned to the closest sampled offset values.



(a) Line (orange) (b) Bin in $\mathcal{HT}$ (c) Filter in $\mathcal{HT}$ (d) $\mathcal{IHT}$

Figure 2.4: Local filters in the Hough domain correspond to global structure in the image domain. (a) An input line in orange. (b) The line becomes a point in Hough domain. (c) A local $[-1,0,1]^\intercal$ filter in Hough domain. (d) The inverse of the local Hough filter corresponds to a global line filter in the image domain.

2.3.2. $\mathcal{IHT}$: FROM HOUGH DOMAIN TO IMAGE DOMAIN

The $\mathcal{HT}$ layer has no learnable parameters, and therefore the gradient is simply a mapping from Hough bins to pixels in the input featuremap, **F**. Following [46], we define the $\mathcal{IHT}$ at pixel location (x, y) as the average of all the bins in $\mathcal{HT}$ where the pixel has voted:

$$\mathcal{IHT}(x,y) = \frac{1}{N_\theta} \sum_\theta \mathcal{HT}(x\cos\theta + y\sin\theta, \theta). \tag{2.4}$$

In the backward pass, $\frac{\partial \mathcal{HT}}{\partial F(x,y)}$, we use equation (2.4) without the normalization over the number of angles, N_θ.

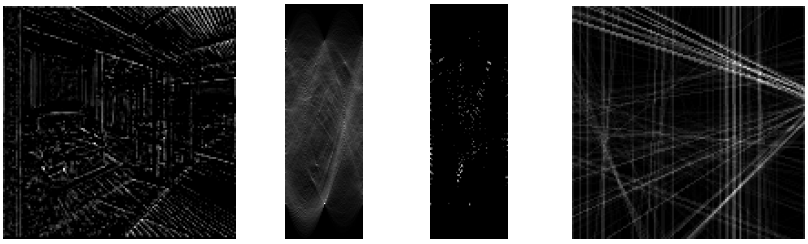
Similar to the forward Hough transform pass, we store the correspondence between the pixels in the input featuremap (x, y) and the Hough transform bins (ρ, θ), in the binary matrix, **B**. We implement the inverse Hough transform as a matrix multiplication of **B** with the learned $\mathcal{HT}$ map, for each channel:

$$\mathcal{IHT} = \mathbf{B}\left(\frac{1}{N_\theta}\mathcal{HT}\right). \tag{2.5}$$

Figure 2.3.(c) shows the $\mathcal{IHT}$ of the Hough transform map in figure 2.3.(b), while figure 2.3.(d) shows the binary mask **B** for the pixel (x', y') highlighted in blue in figure 2.3.(c), mapping it to its corresponding set of bins in the Hough map.

2.3.3. CONVOLUTION IN HOUGH TRANSFORM SPACE

Local operations in Hough space correspond to global operations in the image space, see figure 2.4. Therefore, local convolutions over Hough bins are global convolutions over lines in the image. We learn filters in the Hough domain to take advantage of the global structure, as done in the Radon transform literature [47]. The filtering in the Hough domain is done locally over the offsets, for each angle direction [37, 39]. We perform channel-wise $1D$ convolutions in the Hough space over the offsets, ρ, as the Hough transform is also computed channel-wise over the input featuremaps. In Figure 2.5 we show an example; note that the input featuremap lines are noisy and discontinuous and after applying $1D$ convolutions in Hough space the informative bins are kept and when transformed back to the image domain by the $\mathcal{IHT}$ contains clean lines.



(a) Input featuremap (b) $\mathcal{HT}$ (c) Filtered $\mathcal{HT}$ (d) $\mathcal{IHT}$

Figure 2.5: Noisy local features aggregated globally by learning filters in the Hough domain. (a) Input featuremap with noisy discontinuous lines. (b) The output of the $\mathcal{HT}$ layer using 183 offsets and 60 angles. (c) The result after filtering in the Hough domain. The Hough map contains only the bins corresponding to lines. (d) The output of $\mathcal{IHT}$ layer which receives as input the filtered Hough map. The lines are now clearly visible.

Inspired by the Radon literature [37, 39, 47] we initialize the channel-wise filters, f, with sign-inverted Laplacians by using the second order derivative of a $1D$ Gaussian with randomly sampled scale, σ:

$$f(\rho) \overset{init}{=} -\frac{\partial^2 g(\rho,\sigma)}{\partial \rho^2}, \tag{2.6}$$

where $g(\rho,\sigma)$ is a $1D$ Gaussian kernel. We normalize each filter to have unit L_1 norm and clip it to match the predefined spatial support. We, subsequently, add two more $1D$ convolutional layers for reducing and merging the channels of the Hough transform map. This lowers the computations needed in the inverse Hough transform. Our block is visualized in Figure 2.2.

2.4. EXPERIMENTS

We conduct experiments on three datasets: a controlled Line-Circle dataset, the Wireframe (ShanghaiTech) [5] dataset and the York Urban [29] dataset. We evaluate the added value of global Hough priors, convolutions in the Hough domain, and data efficiency. We provide our source code online[1].

2.4.1. EXP 1: LOCAL AND GLOBAL INFORMATION FOR LINE DETECTION.

Experimental setup. We do a controlled experiment to evaluate the combination of global Hough line priors with learned local features. We target a setting where local-only is difficult and create a Line-Circle dataset of 1,500 binary images of size 100x100 px, split into 744 training, 256 validation, and 500 test images, see figure 2.6. Each image contains 1 to 5 randomly positioned lines and circles of varying sizes. The ground truth has only line segments and we optimize the L_2 pixel difference. We follow the evaluation protocol described in [5, 53, 54] and report AP (average precision) over a number of binarization thresholds varying from 0.1 to 0.9, with a matching tolerance of 0.0075 of the diagonal length [53].

[1] https://github.com/yanconglin/Deep-Hough-Transform-Line-Priors

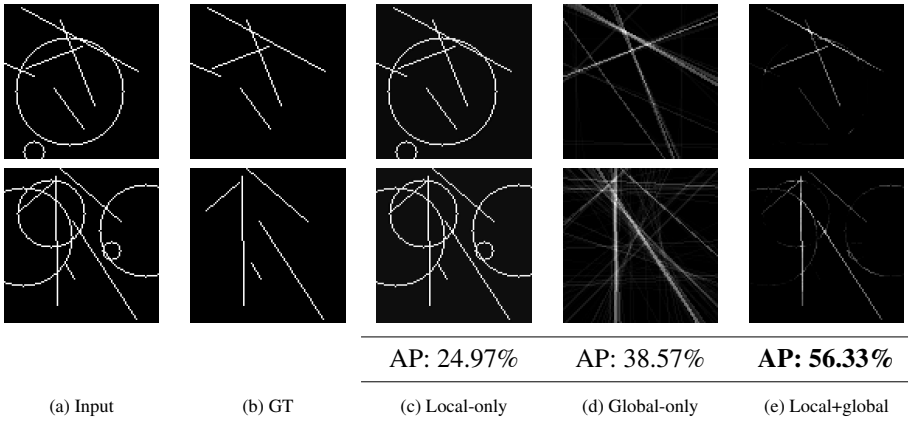


Figure 2.6: **Exp 1:** Results in AP (average precision) and image examples of the Line-Circle dataset. Using local+global information detects not only the direction of the lines, as the global-only does, but also their extent. (See the appendix for more results).

We evaluate three settings: local-only, global-only, and local+global. The aim is not fully solving the toy problem, but rather testing the added value of the $\mathcal{HT}$ and $\mathcal{IHT}$ layers. Therefore, all networks have only 1 layer with 1 filter, where the observed gain in AP cannot be attributed to the network complexity. For local-only we use a a single 3×3 convolutional layer followed by ReLU. For global-only we use an $\mathcal{HT}$ layer followed by a 3×1 convolutional layer, ReLU, and an $\mathcal{IHT}$ layer. For local+global we use the same setting as for global-only, but multiply the output of the $\mathcal{IHT}$ layer with the input image, thus combining global and local image information. All networks have only 1 filter and they are trained from scratch with the same configuration.

Experimental analysis. In the caption of figure 2.6 we show the AP on the Line-Circle dataset. The global-only model can correctly detect the line directions thus it outperforms the local-only model. The global+local model can predict both the line directions and their extent, by combining local and global image information. Local information only is not enough, and indeed the $\mathcal{HT}$ and $\mathcal{IHT}$ layers are effective.

2.4.2. EXP 2: THE EFFECT OF CONVOLUTION IN THE HOUGH DOMAIN

Experimental setup. We evaluate our HT-IHT block design, specifically, the effect of convolutions in the Hough domain on a subset of the Wireframe (ShanghaiTech) dataset [5]. The Wireframe dataset contains 5,462 images. We sample from the training set 1,000 images for training, and 256 images for validation, and use the official test split. As in [7], we resize all images to 512×512 px. The goal is predicting pixels along line segments, where we report AP using the same evaluation setup as in **Exp 1**, and we optimize a binary cross entropy loss.

We use a ResNet [55] backbone architecture, containing 2 convolutional layers with ReLU, followed by 2 residual blocks, and another convolutional layer with a sigmoid activation. The evaluation is done on predictions of 128×128 px, and the ground truth are

Networks	HT-IHT block	AP
(0)	w/o convolution	61.77 %
(1)	$[9 \times 1]$	63.02 %
(2)	$[9 \times 1]$-Laplacian	66.19 %
(3)	$[9 \times 1]$-Laplacian + $[9 \times 1]$ + $[9 \times 1]$	**66.46** %
(4)	$[3 \times 3]$ + $[3 \times 3]$ + $[3 \times 3]$	63.90 %

Table 2.1: **Exp 2:** The effect of convolution in the Hough domain, in terms of AP on a subset of the Wireframe (ShanghaiTech) dataset [5]. No convolutions perform worst (0). The channel-wise Laplacian-initialized filters (2) perform better than the standard $1D$ convolutions (1). Our proposed HT-IHT block (3) versus using $[3 \times 3]$ convolutions (4), shows the added value of following the Radon transform practices.

binary images with line segments. We insert our HT-IHT block after every residual block. All layers are initialized with the He initialization [56].

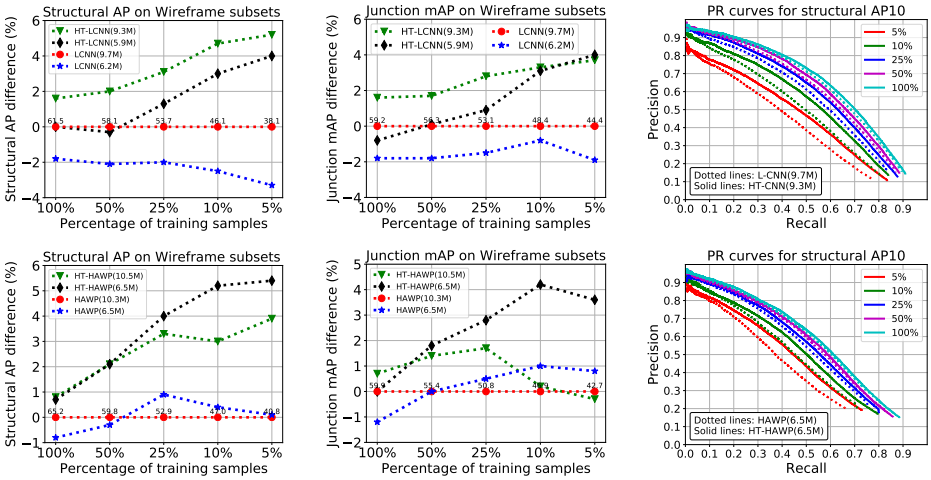
We test the effect of convolutions in the Hough domain by considering in our HT-IHT block: (0) not using any convolutions, (1) using a $1D$ convolution over the offsets, (2) a channel-wise $1D$ convolution initialized with sign-inverted Laplacian filters, (3) our complete HT-IHT block containing Laplacian-initialized $1D$ convolution and two additional $1D$ convolutions, and (4) using three standard 3×3 convolutions.

Experimental analysis. Table 2.1 shows that using convolutions in the Hough domain is beneficial. The channel-wise Laplacian-initialized convolution is more effective than the standard $1D$ convolution using the He initialization [56]. Adding extra convolutions for merging and reducing the channels gives a small improvement in AP, however we use these for practical reasons rather than improved performance. When comparing option (3) with (4), we see clearly the added value of performing $1D$ convolutions over the offsets instead of using standard 3×3 convolutions. This experiment confirms that our choices, inspired from the Radon transform practices, are indeed effective for line detection.

2.4.3. EXP 3: HT-IHT BLOCK FOR LINE SEGMENT DETECTION

Experimental setup. We evaluate our HT-IHT block on the official splits of the Wireframe (ShanghaiTech) [5] and York Urban [29] datasets. We report structural-AP and junction-mAP. Structural-AP is evaluated at AP^5, AP^{10} thresholds, and the junction-mAP is averaged over the thresholds 0.5, 1.0, and 2.0, as in [7]. We also report precision-recall, following [34], which penalizes both under-segmentation and over-segmentation. We use the same distance threshold of $2\sqrt{2}$ px on full-resolution images, as in [34]. For precision-recall, all line segments are ranked by confidence, and the number of top ranking line segments is varied from 10 to 500.

We build on the successful LCNN [6] and HAWP [38] models, where we replace all the hourglass blocks with our HT-IHT block to create HT-LCNN and HT-HAWP, respectively. The hourglass block has twice as many parameters as our HT-IHT block, thus we vary the number of HT-IHT blocks to match the number of parameters of LCNN, HAWP respectively. The networks are trained by the procedure in [7, 38]: optimizing binary cross-entropy loss for junction and line prediction, and L_1 loss for junction offsets. The training



(a) Structural-AP^{10} (b)Junction-mAP (c) PR for structural-AP^{10}

Figure 2.7: **Exp 3.(a):** Data efficiency on subsets of the Wireframe (ShanghaiTech) dataset. We compare different sized variants of our HT-LCNNs and HT-HAWPs with LCNNs [6] and HAWPs [38]. In (a) and (b) we show the absolute difference for structural-AP and junction-mAP compared to the best baseline. In (c) we show PR curves for structural-AP^{10}. Our HT-LCNN and HT-HAWP models are consistently better than their counterparts. The benefit of our HT-IHT block is accentuated for fewer training samples, where with half the number of parameters our models outperform the LCNN and HAWP baselines. Adding geometric priors improves data efficiency.

uses the ADAM optimizer, with scheduled learning rate starting at $4e-4$, and $1e-4$ weight decay, for a maximum of 30 epoch.

EXP 3.(A): EVALUATING DATA EFFICIENCY.

We evaluate data efficiency by reducing the percentage of training samples to $\{50\%, 25\%, 10\%, 5\%\}$ and training from scratch on each subset. We set aside 256 images for validation, and train all the networks on the same training split and evaluate on the official test split. We compare: LCNN(9.7M), LCNN(6.2M) with HT-LCNN(9.3M), HT-LCNN(5.9M), and HAWP(10.3M), HAWP(6.5M) with HT-HAWP(10.5M) and HT-HAWP(6.5M), where we show in brackets the number of parameters.

Figure 2.7 shows structural-AP^{10}, junction-mAP and the PR (precision recall) curve of structural-AP^{10} on the subsets of the Wireframe dataset. Results are plotted relative to our strongest baselines: the LCNN(9.7M) and HAWP(10.3M) models. The HT-LCNN and HT-HAWP models consistently outperform their counterparts. Noteworthy, the HT-LCNN(5.9M) outperforms the LCNN(9.7M) when training on fewer samples, while having 40% fewer parameters. This trend becomes more pronounced with the decrease in training data. We also observe similar improvement for HT-HAWP over HAWP. Figure 2.7(c) shows the PR curve for the structural-AP^{10}. The continuous lines corresponding to HT-LCNN and HT-HAWP are consistently above the dotted lines corresponding to their counterparts, validating that the geometric priors of our HT-IHT block are effective when the amount of training samples is reduced.

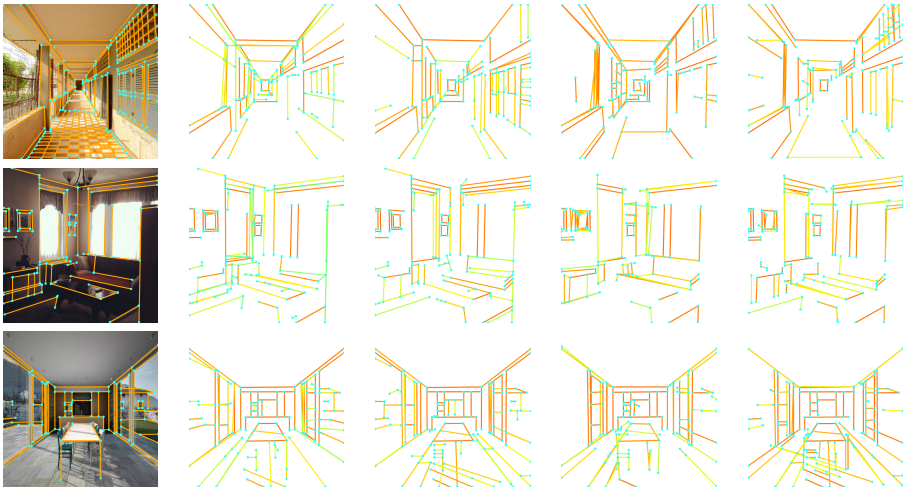


Figure 2.8: **Exp 3.(a):** Visualization of detected wireframes on the Wireframe (ShanghaiTech) dataset, from LCNN(9.7M) and HT-LCNN(9.3M) trained on 100% and 10% data subsets. HT-LCNN can more consistently detects the wireframes, when trained on 10% subset, compared to LCNN. (See the appendix for more results).

Figure 2.8 visualizes top 100 line-segment predictions of LCNN (9.7M) and HT-LCNN (9.3M) trained on 100% and 10% subsets of the Wireframe dataset. When comparing the LCNN and HT-LCNN in the top row, we notice that HT-LCNN is more precise, especially when training on only 10% of the data. HT-LCNN detects more lines and junctions than LCNN because it identifies lines as local maxima in the Hough space. HT-LCNN relies less on contextual information, and thus it predicts all possible lines as wireframes (e.g. shadows of objects in the third row). In comparison, L-CNN correctly ignores those line segments. Junctions benefit from more lines, as they are intersections of lines. These results shows the added value of HT-LCNN when training on limited data.

EXP 3.(B): COMPARISON WITH STATE-OF-THE-ART.

We compare our HT-LCNN and HT-HAWP, starting from LCNN [6] and HAWP [38] and using HT-IHT blocks instead of the hourglass blocks, with five state-of-the-art models on the Wireframe (ShanghaiTech) [5] and York Urban [29] datasets. The official training split of the Wireframe dataset is used for training, and we evaluate on the respective test splits of the Wireframe/York Urban datasets. We consider three methods employing knowledge-based features: LSD [28], Linelet [26] and MCMLSD [34], and four learning-based methods: AFM [13], WF-Parser (Wireframe Parser) [5], LCNN [6], HAWP [38]. We use the pre-trained models provided by the authors for AFM, LCNN and HAWP, while the WF-Parser, HT-LCNN, and HT-HAWP are trained from scratch by us.

Table 2.2 compares structural-AP^5, -AP^{10} and junction-mAP for seven state-of-the-art methods. We report the number of parameters for the learning-based models as well as the frames per second (FPS) measured by using a single CPU thread or a single GPU (GTX 1080 Ti) over the test set. Our models using the HT-IHT block outperform existing methods

Train/test			Wireframe / Wireframe			Wireframe / York Urban		
			Structural		Junction	Structural		Junction
Metrics	#Params	FPS	AP^5	AP^{10}	mAP	AP^5	AP^{10}	mAP
LSD [28]	—	15.3	7.1	9.3	16.5	7.5	9.2	14.9
Linelet [26]	—	0.04	8.3	10.9	17.4	9.0	10.8	18.2
MCMLSD [34]	—	0.2	7.6	10.4	13.8	7.2	9.2	14.8
WF-Parser [5]	31 M	1.7	6.9	9.0	36.1	2.8	3.9	22.5
AFM [13]	43 M	6.5	18.3	23.9	23.3	7.1	9.1	12.3
LCNN [6]	9.7 M	10.8	58.9	62.9	59.3	24.3	26.4	30.4
HT-LCNN (Our)	9.3 M	7.5	60.3	64.2	60.6	25.7	28.0	**32.5**
HAWP [38]	10.3 M	13.6	62.5	66.5	60.2	**26.1**	**28.5**	31.6
HT-HAWP (Our)	10.5 M	12.2	**62.9**	**66.6**	**61.1**	25.0	27.4	31.5

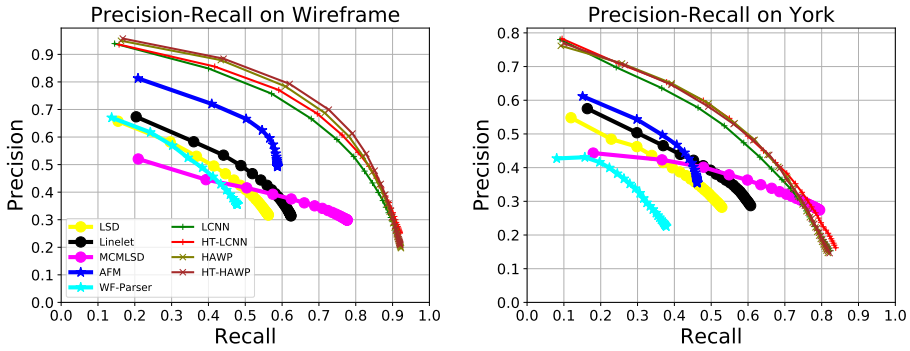
Table 2.2: **Exp 3.(b):** Comparing state-of-the-art line detection methods on the Wireframe (ShanghaiTech) and York Urban datasets. We report the number of parameters and FPS timing for every method. Our HT-LCNN and HT-HAWP using HT-IHT blocks, show competitive performance. HT-HAWP is similar to HAWP on the Wireframe dataset, while being less precise on the York Urban dataset. When compared to LCNN, our HT-LCNN consistently outperforms the baseline, illustrating the added value of the Hough priors.

on the Wireframe dataset, and show rivaling performance on the York Urban dataset. HT-HAWP performs similar to HAWP on the Wireframe dataset while being less competitive on the York Urban dataset. HAWP uses a proposal refinement module, which further removes unmatched line proposals. This dampens the advantage of our HT-IHT block. Given that the York Urban dataset is not fully annotated, this may negatively affect the performance of our HT-IHT block. However, adding HT-IHT block improves the performance of HT-LCNN over LCNN on both datasets, which shows the added value of the geometric line priors. Moreover, HAWP and LCNN perform well when ample training data is available. When limiting the training data, their performances decrease by a large margin compared with our models, as exposed in **Exp 3.(a)**.

Figure 2.9 shows precision-recall scores [34] on the Wireframe (ShanghaiTech) and York Urban datasets. MCMLSD [34] shows good performance in the high-recall zone on the York Urban dataset, but its performance is lacking in the low-recall zone. AFM [13] predicts a limited number of line segments, and thus it lacks in the high-recall zone. One advantage of (HT-)LCNN and (HT-)HAWP over other models such as AFM, is their performance in the high-recall zone, indicating that they can detect more ground truth line segments. However, they predict more overlapping line segments due to co-linear junctions, which results in a rapid decrease in precision. Our proposed HT-LCNN and HT-HAWP show competitive performance when compared to state-of-the-art models, thus validating the usefulness of the HT-IHT block.

In figure 2.10, we compare our HT-LCNN and HT-HAWP with PPGNet [41]. The PPGNet result is estimated from the original paper, since we are not able to replicate the results using the author's code [2]. We follow the same protocol as PPGNet to evaluate (HT-)LCNN and (HT-)HAWP. In general, PPGNet shows superior performance on the York

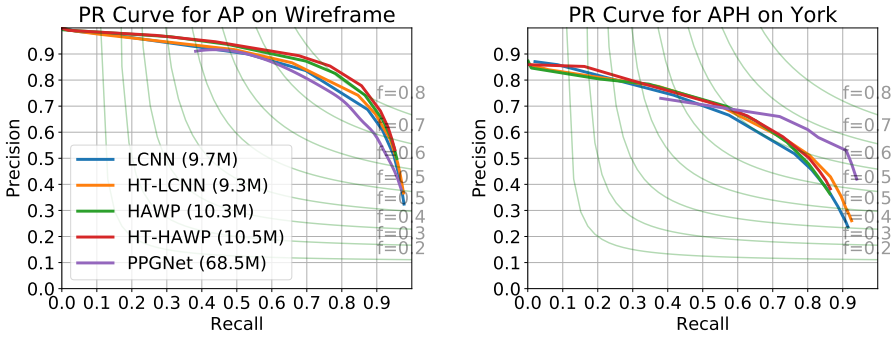
[2] https://github.com/svip-lab/PPGNet



(a) Precision-recall on Wireframe (ShanghaiTech)

(b) Precision-recall on York Urban

Figure 2.9: **Exp 3.(b):** Comparing our HT-LCNN and HT-HAWP with seven existing methods using precision-recall scores on the Wireframe (ShanghaiTech) and York Urban datasets. Traditional knowledge-based methods are outperformed by deep learning methods. Among the learning-based methods, our proposed HT-LCNN and HT-HAWP achieve state-of-the-art performance, even in the full-data regime.



(a) AP on Wireframe (ShanghaiTech)

(b) AP on York Urban

Figure 2.10: **Exp 3.(b):** Comparing PPGNet[41] with (HT-)LCNN and (HT-)HAWP on the Wireframe (ShanghaiTech) and York Urban datasets. PPGNet shows better performance on the York Urban dataset, especially in high-recall region, while being slightly less precise on the Wireframe dataset when compared to our HT-LCNN and HT-HAWP methods. We show between brackets the number of parameters.

Urban dataset, especially in the high-recall region, while using a lot more parameters. However, our HT-LCNN and HT-HAWP methods are slightly more precise on the Wireframe dataset.

2.5. CONCLUSION

We propose adding geometric priors based on Hough transform, for improved data efficiency. The Hough transform priors are added end-to-end in a deep network, where we detail the forward and backward passes of our proposed HT-IHT block. We additionally introduce the use of convolutions in the Hough domain, which are effective at retaining only the line information. We demonstrate experimentally on a toy Line-Circle dataset that

our $\mathcal{HT}$ (Hough transform) and $\mathcal{IHT}$ (inverse Hough transform) layers, inside the HT-IHT block, help detect lines by combining local and global image information. Furthermore, we validate on the Wireframe (ShanghaiTech) and York Urban datasets that the Hough line priors, included in our HT-IHT block, are effective when reducing the training data size. Finally, we show that our proposed approach achieves competitive performance when compared to state-of-the-art methods.

2.6. APPENDIX

2.6.1. EXP 1: QUALITATIVE RESULTS ON THE LINE-CIRCLE DATASET

Figure 2.11 visualizes detected lines on the Line-Circle dataset from the local-only, global-only and local+global models. Using the global information learned by our HT-IHT block combined with the local information provided by the convolutional layers, we propose a local+global approach that can predict both the direction of the lines and their extent.

2.6.2. EXP 3.(A): QUALITATIVE RESULTS USING WIREFRAME SUBSETS

Figure 2.12 visualizes detected wireframes from our HT-LCNN (9.3M) and LCNN (9.7M) [6] trained on various Wireframe subsets [5]. We display the top 100 line segments. In the first example, our HT-LCNN is better than LCNN in detecting wireframes of windows on various subsets. However, our HT-LCNN is not able to ignore the shadow of objects, compared to LCNN, as shown in the last example. In general, HT-LCNN outperforms LCNN when training data is limited.

2.6.3. EXP 3.(B): QUALITATIVE COMPARISON WITH THE STATE-OF-THE-ART ON THE WIREFRAME DATASET

Figure 2.14 visualizes detected line segments from different approaches on the Wireframe dataset [5]. We follow [13] to set up thresholds for LSD [28] and WF-Parser [5], and select the top 100 line segments for other methods (HT-HAWP, HT-LCNN, HAWP[38], LCNN [6], AFM [13], MCMLSD [34] and Linelet [26].) Learning-based models predict line segments more precisely than the non-learning methods. In general, our models with HT-IHT block perform competitively with the state-of-the-art.

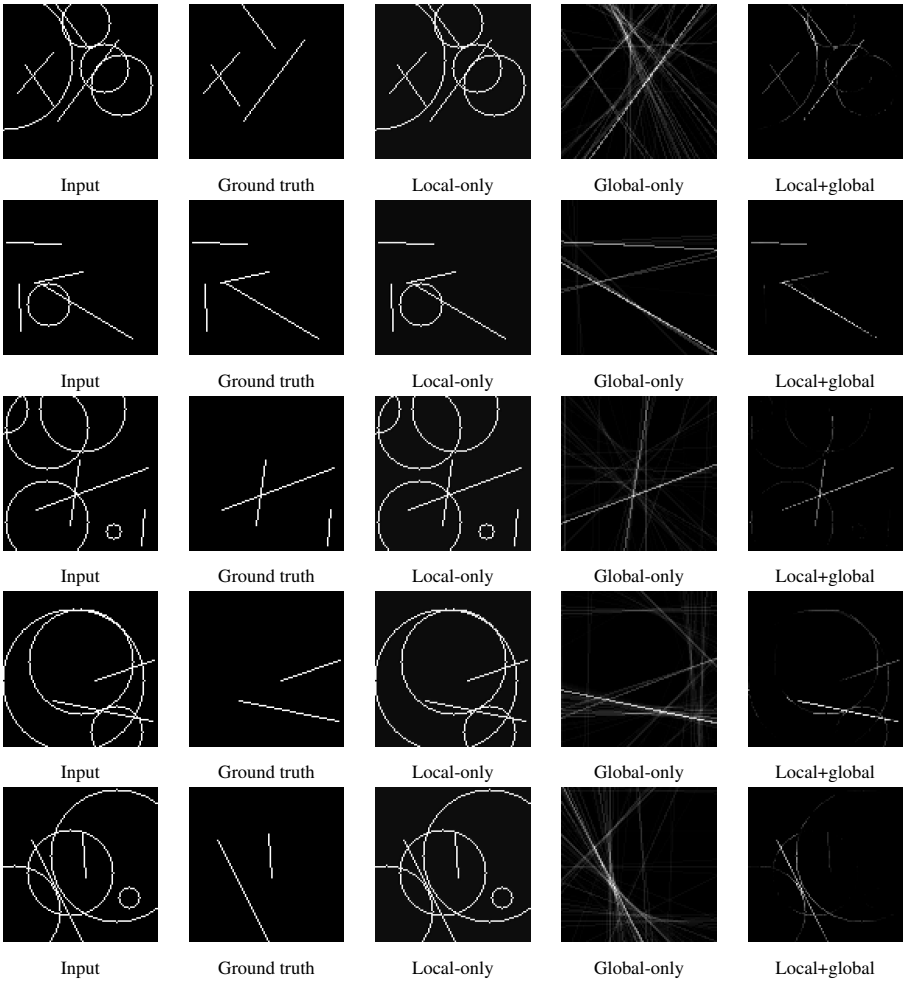


Figure 2.11: **Exp 1:** Visualization of detected lines on the toy Line-Circle dataset. The local+global model successfully removed the circle pixels and retains the pixels along the line. Combing local and global information detects not only the direction of the lines but also their extent.

Figure 2.12: **Exp 3.(a):** Visualization of detected wireframes from HT-LCNN (9.3M) and LCNN (9.7M) [6] trained on various Wireframe subsets [5]. Our HT-LCNN can more precisely detect the wireframes of the windows than LCNN, as shown in the first example. However, our HT-LCNN generates more false-positive predictions from the shadow of objects, when compared to LCNN, as shown in the last example.

Figure 2.13: **Exp 3.(b):** Visualization of detected line segments on the Wireframe dataset [5]. We show predictions from our HT-HAWP, HT-LCNN, and seven other leading methods: HAWP[38], LCNN [6], AFM [13], WF-Parser [5], MCMLSD [34], Linelet [26] and LSD [28]). (Continued on the next page.)

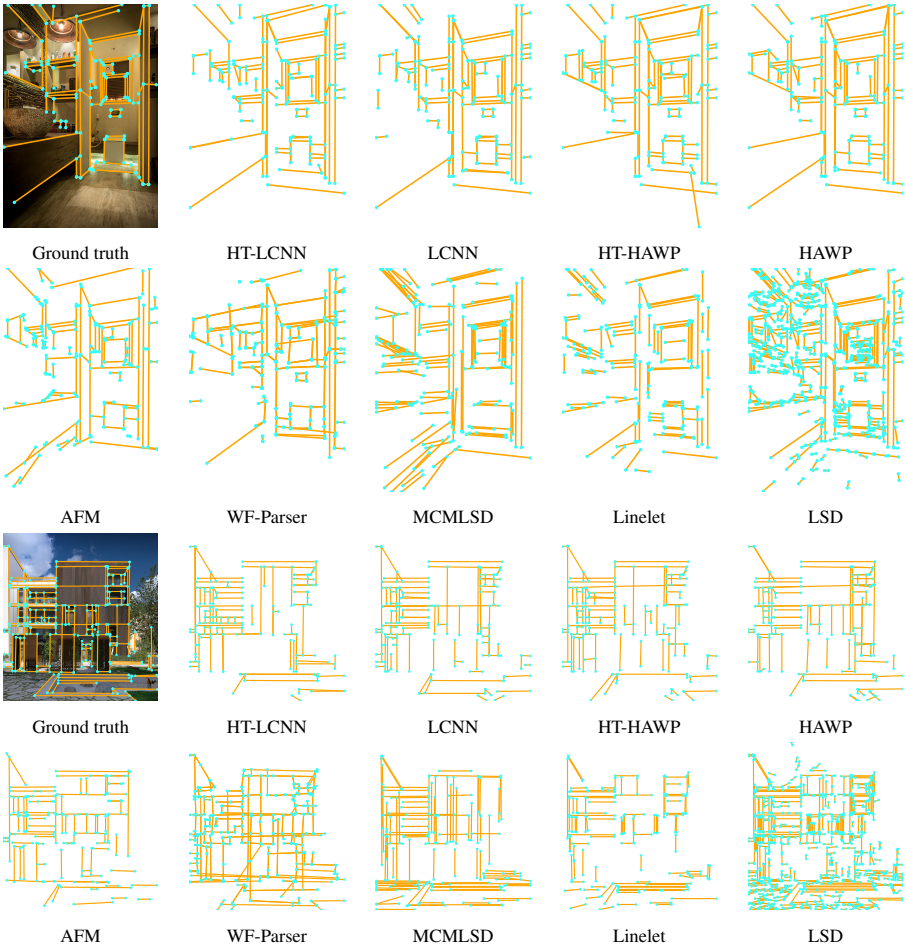


Figure 2.14: **Exp 3.(b):** Visualization of detected wireframes (line segments) on the Wireframe dataset [5]. We show predictions from our HT-HAWP, HT-LCNN and seven other leading methods (HAWP[38], LCNN [6], AFM [13], WF-Parser [5], MCMLSD [34], Linelet [26] and LSD [28]). In general, learning-based methods are able to detect line segments more precisely, while MCMLSD, Linelet and LSD generate more false-positive predictions. The HT-LCNN and HT-HAWP predictions preserve both global structures and local details, and show competitive performance with the leading methods.

REFERENCES

[1] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, *Recent progress in road and lane detection: a survey,* Machine vision and applications **25**, 727 (2014).

[2] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon, *Vpgnet: Vanishing point guided network for lane and road marking detection and recognition,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 1947–1955.

[3] J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao, *Robust lane detection using two-stage feature extraction with curve fitting,* Pattern Recognition **59**, 225 (2016).

[4] R. K. Satzoda and M. M. Trivedi, *Efficient lane and vehicle detection with integrated synergies (elvis),* in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014) pp. 708–713.

[5] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, *Learning to parse wireframes in images of man-made environments,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 626–635.

[6] Y. Zhou, H. Qi, and Y. Ma, *End-to-end wireframe parsing,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 962–971.

[7] Y. Zhou, H. Qi, Y. Zhai, Q. Sun, Z. Chen, L.-Y. Wei, and Y. Ma, *Learning to reconstruct 3d manhattan wireframes from a single image,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 7698–7707.

[8] E. Gershikov, T. Libe, and S. Kosolapov, *Horizon line detection in marine images: which method to choose?* International Journal on Advances in Intelligent Systems **6** (2013).

[9] L. Porzi, S. Rota Bulò, and E. Ricci, *A deeply-supervised deconvolutional network for horizon line detection,* in *Proceedings of the 24th ACM international conference on Multimedia* (2016) pp. 137–141.

[10] G. Simon, A. Fond, and M.-O. Berger, *A-contrario horizon-first vanishing point detection using second-order grouping laws,* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018) pp. 318–333.

[11] J. Sun, L. Liang, F. Wen, and H.-Y. Shum, *Image vectorization using optimized gradient meshes,* ACM Transactions on Graphics (TOG) **26**, 11 (2007).

[12] J. J. Zou and H. Yan, *Cartoon image vectorization based on shape subdivision,* in *Proceedings. Computer Graphics International 2001* (2001) pp. 225–231.

[13] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, *Learning attraction field representation for robust line segment detection,* in *The IEEE Conference on Computer Vision and Pattern Recognition* (2019).

[14] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, *Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models,* in *Advances in Neural Information Processing Systems* (2019) pp. 9448–9458.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge,* International Journal of Computer Vision (IJCV) **115**, 211 (2015).

[16] J. Bruna and S. Mallat, *Invariant scattering convolution networks,* IEEE transactions on pattern analysis and machine intelligence **35**, 1872 (2013).

[17] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. Smeulders, *Structured receptive fields in cnns,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 2610–2619.

[18] O. S. Kayhan and J. C. van Gemert, *On translation invariance in cnns: Convolutional layers can exploit absolute spatial location,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 14274–14285.

[19] G. Urban, K. J. Geras, S. E. Kahou, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson, *Do deep convolutional nets really need to be deep and convolutional?* International Conference on Learning Representations (2016).

[20] E. Shelhamer, D. Wang, and T. Darrell, *Blurring the line between structure and learning to optimize and adapt receptive fields,* CoRR (2019).

[21] I. Sosnovik, M. Szmaja, and A. Smeulders, *Scale-equivariant steerable networks,* International Conference on Learning Representations (2020).

[22] R. O. Duda and P. E. Hart, *Use of the hough transformation to detect lines and curves in pictures,* Communications of the ACM **15**, 11 (1972).

[23] V. Kamat-Sadekar and S. Ganesan, *Complete description of multiple line segments using the hough transform,* Image and Vision Computing **16**, 597 (1998).

[24] J. Matas, C. Galambos, and J. Kittler, *Robust detection of lines using the progressive probabilistic hough transform,* Computer Vision and Image Understanding **78**, 119 (2000).

[25] J. B. Burns, A. R. Hanson, and E. M. Riseman, *Extracting straight lines,* IEEE transactions on pattern analysis and machine intelligence , 425 (1986).

[26] N.-G. Cho, A. Yuille, and S.-W. Lee, *A novel linelet-based representation for line segment detection,* IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 1195 (2017).

[27] V. Pătrăucean, P. Gurdjos, and R. G. Von Gioi, *A parameterless line segment and elliptical arc detector with enhanced ellipse fitting,* in *European Conference on Computer Vision* (2012) pp. 572–585.

[28] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *Lsd: A fast line segment detector with a false detection control,* IEEE Transactions on Pattern Analysis and Machine Intelligence **32**, 722 (2008).

[29] P. Denis, J. H. Elder, and F. J. Estrada, *Efficient edge-based methods for estimating manhattan frames in urban imagery,* in *European Conference on Computer Vision* (Springer, 2008) pp. 197–210.

[30] Y. Furukawa and Y. Shinagawa, *Accurate and robust line segment extraction by analyzing distribution around peaks in hough space,* Computer Vision and Image Understanding **92**, 1 (2003).

[31] Z. Xu, B.-S. Shin, and R. Klette, *Accurate and robust line segment extraction using minimum entropy with hough transform,* IEEE Transactions on Image Processing **24**, 813 (2014).

[32] Z. Xu, B.-S. Shin, and R. Klette, *A statistical method for line segment detection,* Computer Vision and Image Understanding **138**, 61 (2015).

[33] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *On straight line segment detection,* Journal of Mathematical Imaging and Vision **32**, 313 (2008).

[34] E. J. Almazan, R. Tal, Y. Qian, and J. H. Elder, *Mcmlsd: A dynamic programming approach to line segment detection,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 2031–2039.

[35] R. F. Guerreiro and P. M. Aguiar, *Connectivity-enforcing hough transform for the robust extraction of line segments,* IEEE Transactions on Image Processing **21**, 4819 (2012).

[36] A. Sheshkus, A. Ingacheva, V. Arlazarov, and D. Nikolaev, *Houghnet: neural network architecture for vanishing points detection,* International Conference on Document Analysis and Recognition (ICDAR) (2019).

[37] D. P. Nikolaev, S. M. Karpenko, I. P. Nikolaev, and P. P. Nikolayev, *Hough transform: underestimated tool in the computer vision field,* in *Proceedings of the 22th European Conference on Modelling and Simulation*, Vol. 238 (2008) p. 246.

[38] N. Xue, T. Wu, S. Bai, F. Wang, G.-S. Xia, L. Zhang, and P. H. Torr, *Holistically-attracted wireframe parsing,* in *Conference on Computer Vision and Pattern Recognition* (2020).

[39] H. Wei, W. Bing, and Z. Yue, *X-linenet: Detecting aircraft in remote sensing images by a pair of intersecting line segments,* CoRR (2019).

[40] V. N. Nguyen, R. Jenssen, and D. Roverso, *Ls-net: Fast single-shot line-segment detector,* CoRR (2019).

[41] Z. Zhang, Z. Li, N. Bi, J. Zheng, J. Wang, K. Huang, W. Luo, Y. Xu, and S. Gao, *Ppgnet: Learning point-pair graph for line segment detection,* in *Conference on Computer Vision and Pattern Recognition* (2019).

[42] S. Workman, M. Zhai, and N. Jacobs, *Horizon lines in the wild,* British Machine Vision Conference (2016).

[43] M. C. Beltrametti, C. Campi, A. M. Massone, and M.-L. Torrente, *Geometry of the hough transforms with applications to synthetic data,* CoRR (2019).

[44] C. R. Qi, O. Litany, K. He, and L. J. Guibas, *Deep hough voting for 3d object detection in point clouds,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 9277–9286.

[45] J. Min, J. Lee, J. Ponce, and M. Cho, *Hyperpixel flow: Semantic correspondence with multi-layer neural features,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 3395–3404.

[46] J. Beatty, *The Radon Transform and the Mathematics of Medical Imaging*, Honors thesis (Digital Commons @ Colby, 2012).

[47] M. Magnusson, *Linogram and Other Direct Fourier Methods for Tomographic Reconstruction*, Linköping studies in science and technology: Dissertations (Department of Mechanical Engineering, Linköping University, 1993).

[48] P. Toft, *The Radon Transform: Theory and Implementation* (Department of Mathematical Modelling, Section for Digital Signal Processing, Technical University of Denmark, 1996).

[49] J. He and J. Ma, *Radon inversion via deep learning,* in *Medical Imaging* (2018).

[50] D. Rim, *Exact and fast inversion of the approximate discrete radon transform from partial data,* Applied Mathematics Letters **102**, 106159 (2020).

[51] M. N. Do and M. Vetterli, *The finite ridgelet transform for image representation,* IEEE Transactions on image Processing **12**, 16 (2003).

[52] Q. Wei, D. Feng, and W. Zheng, *Funnel transform for straight line detection,* CoRR (2019).

[53] D. R. Martin, C. C. Fowlkes, and J. Malik, *Learning to detect natural image boundaries using local brightness, color, and texture cues,* IEEE transactions on pattern analysis and machine intelligence **26**, 530 (2004).

[54] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, *Using contours to detect and localize junctions in natural images,* in *2008 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2008) pp. 1–8.

[55] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[56] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,* in *Proceedings of the IEEE international conference on computer vision* (2015) pp. 1026–1034.

3

SEMI-SUPERVISED LANE DETECTION WITH DEEP HOUGH TRANSFORM

Current work on lane detection relies on large manually annotated datasets. We reduce the dependency on annotations by leveraging massive cheaply available unlabelled data. We propose a novel loss function exploiting geometric knowledge of lanes in Hough space, where a lane can be identified as a local maximum. By splitting lanes into separate channels, we can localize each lane via simple global max-pooling. The location of the maximum encodes the layout of a lane, while the intensity indicates the the probability of a lane being present. Maximizing the log-probability of the maximal bins helps neural networks find lanes without labels. On the CULane and TuSimple datasets, we show that the proposed Hough Transform loss improves performance significantly by learning from large amounts of unlabelled images.

This chapter is published as:
Yancong Lin, Silvia-Laura Pintea, and Jan van Gemert. *Semi-Supervised Lane Detection with Deep Hough Transform*. International Conference on Image Processing (ICIP), 2021. arXiv: https://arxiv.org/abs/2106.05094.

3.1. INTRODUCTION

One key component of self-driving cars is the lane-keeping assist [1, 2], which actively keeps the vehicles in the marked lanes. The lane-keeping assist relies on accurate lane detection in the wild, which is a highly challenging task because of illumination and appearance variations, traffic flow, and new unseen driving scenarios [3].

State-of-the-art deep learning methods for lane detection perform remarkably well on benchmark datasets [3–6]. However, they rely on deep networks powered by massive amounts of labelled data. Although the data itself can be obtained at relatively low cost, it's their annotations that are laborious and thus expensive [7]. Moreover, the existing curated datasets do not cover all the possible driving scenarios that could be encountered in real-world situations. Being able to leverage additional realistic unlabelled training data would allow for a more robust lane detection system.

To make effective use of additional unlabelled data, we propose a semi-supervised Hough Transform-based loss which exploits geometric prior knowledge of lanes in the Hough space [8, 9].

Lanes are lines, thus we propose a semi-supervised Hough Transform loss that parameterizes lines in Hough space, by mapping them to individual bins represented by an offset and an angle. Inspired by the work in [9], we rely on a trainable Hough Transform and Inverse Hough Transform (HT-IHT) module embedded into a neural network to learn Hough representations for lane detection. We subsequently extend its use for semi-supervised training, by noting that the presence of lanes leads to Hough bins with maximal votes. Maximizing the log-probability of these Hough bins requires no human supervision, enabling the network to detect lanes in unlabelled images.

This paper makes the following contributions: (1) we present an annotation-efficient approach for lane detection in a semi-supervised way; (2) to this end, we propose a novel loss function to exploit prior geometric knowledge of lanes in Hough space; (3) we experimentally show improved performance on the CULane [10] and TuSimple [11] datasets, given large amounts of unlabelled data.

3.2. RELATED WORK

Lane detection methods. Classic work on lane detection is based on knowledge-based manually designed geometric features. Examples include grouping image gradients [12–15], or line detection techniques through Hough Transform [16–19] relying on local edges extracted using image gradients. A main drawback of such knowledge-based methods is their inability to handle complex scenarios where traffic flow and illumination conditions change dramatically. Here, we address this by learning appearance variation of lanes in a deep network, while still relying on the Hough Transform as prior knowledge for line detection [8, 9].

Recently, deep neural networks have been employed for efficient lane detection, replacing well-engineered features. Typically, the learning-based methods treat the lane detection as a semantic segmentation task and learn semantic features from large datasets [1, 20–24]. In contrast to these works we improve the prediction accuracy by leveraging massive unlabelled data through semi-supervised learning.

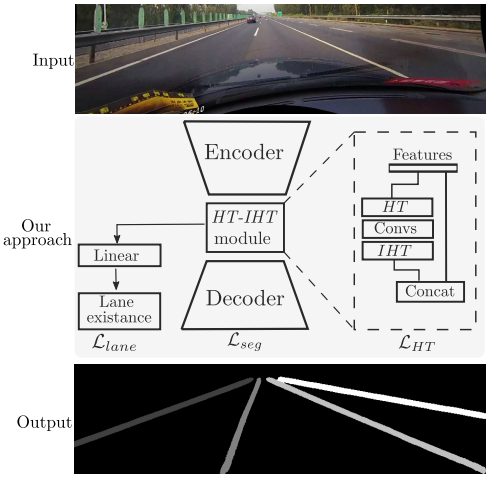


Figure 3.1: **Overview of our model.** We have an encoder, a decoder, and a fully connected layer inspired by the ERFNet [4, 5] with a trainable Hough-Transform (HT) and Inverse Hough-Transform (IHT) module [9], on top of which we build our HT-based semi-supervised loss maximizing the probability of the maximal bins in Hough space, where $\mathcal{L}_{lane}$, $\mathcal{L}_{seg}$, and $\mathcal{L}_{HT}$ are the optimized loss functions.

Semi-supervised methods. Semi-supervised methods solve the learning task by relying on both labelled and unlabelled data [25], and are divided into: inductive approaches constructing a classifier over labelled and unlabelled data [26–28], and transductive approaches propagating where the task information is shared between data points [29–31]. A self-driving car has no access to the test statistics, therefore we consider the inductive case.

3.3. SEMI-SUPERVISED LANE DETECTION

Given an input image, our model outputs a lane probability and a semantic segmentation mask of lane pixels. We use as a starting point the popular ERFNet [5][1]. The ERFNet contains a convolutional encoder for deep feature extraction, a convolutional decoder for lane predictions, and a fully connected layer for predicting the probability of a lane. We insert a trainable Hough Transform and Inverse Hough Transform (*HT-IHT*) block [9] between the encoder and decoder, and utilize the Hough representations of lanes for semi-supervised learning. Figure 6.2 depicts the overall structure of our model.

3.3.1. HOUGH TRANSFORM LINE PRIORS

We encode an input image to a semantic feature representation F which is mapped to the Hough space, through a trainable Hough Transform module [9]. The Hough transform HT maps a feature map F of size $[H \times W]$ to an $[N_\rho \times N_\theta]$ Hough histogram, where N_ρ and N_θ are the number of discrete offsets and angles. Pixels along lines in F are mapped into discrete pairs of offsets ρ and angles θ. Specifically, given a line indexed by its pixels

[1] https://github.com/cardwing/Codes-for-Lane-Detection

(x_i, y_i), they all vote in the Hough space for the closest bin (ρ, θ):

$$HT(\rho,\theta) = \sum_i F(x_i, y_i), \tag{3.1}$$

where the mapping is given by $\rho = x_i \cos\theta + y_i \sin\theta$.

We perform a set of 1D convolutions in Hough space over the offset direction and apply an Inverse Hough Transform IHT module mapping the $[N_\rho \times N_\theta]$ Hough histogram back to an $[H \times W]$ feature map [9]. The IHT maps bins (ρ, θ) to pixels (x_i, y_i) by averaging all the HT bins where a certain pixel has voted:

$$IHT(x_i, y_i) = \frac{1}{N_\theta} \sum_\theta HT(x_i \cos\theta + y_i \sin\theta, \theta). \tag{3.2}$$

We concatenate the features F with the IHT features, followed by a convolutional layer merging these two branches. We set $H = 26$, $W = 122$, $N_\rho = 125$ and $N_\theta = 60$.

3.3.2. HOUGH TRANSFORM LOSS FOR UNLABELLED DATA

A lane is composed of a set of line segments with a certain width, that share the same orientation. For unlabelled images we rely on the observation that lanes correspond to local maxima in the Hough space. Since the ERFNet [4, 5] predicts a single lane in each output channel, the mapping to Hough space recovers the lanes as global maxima in their respective channels. Having a large global maximum indicates that pixels along that line direction are well aligned, thus falling in the same bin. Based on this observation, we provide supervision to unlabelled inputs by maximizing the log-probability of the maximum bin $(\hat{\rho}, \hat{\theta})$ in Hough domain. To give the HT bins a probabilistic interpretation, we rescale the HT maps between $[0, 1]$ for each angle direction independently by applying an L_1 normalization over the offset dimension:

$$\mathcal{L}_{HT} = -\log\left(\frac{HT(\hat{\rho},\hat{\theta})}{\sum_{k=0}^{N_\rho} HT(\rho_k,\hat{\theta})}\right), \tag{3.3}$$

where $(\hat{\rho}, \hat{\theta})$ is the positions of the global maximum in Hough space, calculated from the predicted segmentation masks.

3.3.3. TRAINING WITH BOTH LABELLED AND UNLABELLED DATA

We train our model with both labelled and unlabelled data. As in [3, 4] the network predicts for each channel a mask used in a cross entropy loss $\mathcal{L}_{\text{seg}}$ over labelled data for predicting the semantic segmentation. Additionally, the network predicts lane probabilities p which are used in a binary cross entropy loss over labelled data $\mathcal{L}_{\text{lane}}$ for optimizing for the existence of a lane. We also optimize the proposed $\mathcal{L}_{HT}$, only when the predicted probability p of a lane is larger than a threshold τ; otherwise, we skip the corresponding lane. We set $\tau = 0.9$. The total loss is the combination of the three losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{seg}} + \alpha\mathcal{L}_{\text{lane}} + \beta\mathcal{L}_{HT}(p, \tau), \tag{3.4}$$

where α and β are used to balance different loss terms.

3.4. EXPERIMENTAL ANALYSIS

Datasets. We evaluate our models on the TuSimple dataset [11] and CULane dataset [10]. All video clips in TuSimple dataset are taken on highways. There are 3,626 frames for training and 2,782 frames for testing. The CULane dataset contains images from 9 different driving scenarios, such as lanes in shadow and at night with poor lighting conditions. There are 88,880 images for training, 9,675 for validation, and 34,680 images for testing. We follow the official evaluation protocol to measure accuracy on the TuSimple, and use F_1 measure on the CULane dataset.

Baselines. We compare with the baseline ERFNet [5], and with the ERFNet-HT using the *HT-IHT* block [9]. Both models are trained from scratch with labelled data only. For semi-supervised learning, we consider the pseudo-labeling baseline ERFNet-$pseudo$, and our proposed ERFNet-HT-$\mathcal{L}_{HT}$. The ERFNet-$pseudo$ baseline first learns to predict lanes on annotated data only, and subsequently uses the predicted pseudo-labels to annotate unlabelled data, and then retrains the model on all data. ERFNet-$pseudo$ treats the prediction with a confidence score larger than 0.9 as "ground truth" and optimize the $\mathcal{L}_{\text{seg}}$ with pseudo-labels. ERFNet-HT-$\mathcal{L}_{HT}$ uses our proposed $\mathcal{L}_{HT}$ loss. ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ combines both pseudo-labelling and the $\mathcal{L}_{HT}$ loss. Additionally, we also compare with s4GAN [32], a state-of-the-art semi-supervised learning model for semantic segmentation.

Implementation details. We follow the implementation in [4] and provide our code online [2]. We use SGD [33] to train ERFNet and ERFNet-HT for 24 epochs. ERFNet-$pseudo$, ERFNet-HT-$\mathcal{L}_{HT}$ and ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ are trained with extra unlabelled data for another 12 epochs. The initial learning rate is 1×10^{-2}, and is decreased by a factor of $(1 - t/T)^{0.9}$, where t is the current training epoch and T is the total number of epochs, as in [4]. The batch size is set to be 16. For our $\mathcal{L}_{\text{total}}$, we set the weights $\alpha = 0.1$ and $\beta = 0.01$ to ensure that all loss terms have similar magnitudes. Following [4], we multiply the $\mathcal{L}_{\text{seg}}$ for the background class by 0.4 to counter the large number of background pixels. For s4GAN [32], we directly use the official implementation [3].

Results analysis. To evaluate the effectiveness of our $\mathcal{L}_{HT}$ in utilizing unlabelled data, we randomly split the CULane training data into $\{100/0, 50/50, 10/90, 5/95, 1/99\}$ sets, where the first digit indicates the proportion of labelled data, while the second one is the proportion of unlabelled data. The TuSimple dataset is split into $\{100/0, 50/50, 10/90\}$ sets, as it contains only 3,626 images. We use the same splits for all models. We report accuracy on TuSimple and F_1-measure on the CULane dataset.

Table 3.1 compares all models on various training sets. ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ achieves the best performance on both 50% and 10% subsets of TuSimple dataset. The improvement over the supervised baseline is more than 15% on the 10% subset. All semi-supervised ERFNet models improve accuracy, indicating the potential of exploiting massive unlabelled data. Pseudo-labeling allows learning from high confidence predictions explicitly, while $\mathcal{L}_{HT}$ optimizes line feature representations in Hough space in an implicit way. However, s4GAN [32] shows inferior performance to other models, due to the fact that s4GAN is not specifically optimized for lane detection, where image content differs sub-

[2] https://github.com/yanconglin/Semi-Supervised-Lane-Detection-with-Deep-Hough-Transform

[3] https://github.com/sud0301/semisup-semseg

Table 3.1: **Performance on TuSimple and CULane datasets with various amounts of labelled and unlabelled data.** The first column indicates the proportion of labelled data for training. The remaining data is treated as unlabelled for semi-supervised learning. ERFNet-HT-$\mathcal{L}_{HT}$ and ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ show performance improvements on both datasets. When the number of labelled samples decreases, the advantage of ERFNet-HT-$\mathcal{L}_{HT}$ is more pronounced.

Labels	s4GAN [32]	ERFNet models				
		Baseline [5]	HT[9]	$pseudo$	HT-$\mathcal{L}_{HT}$	HT-$pseudo$ + $\mathcal{L}_{HT}$
		Accuracy (%) on the TuSimple dataset				
100%	-	**93.71**	**93.71**	-	-	-
50%	88.82	92.97	93.47	93.37	93.63	**93.70**
10%	86.25	82.97	77.71	92.12	92.98	**93.05**
		F_1 scores on the CULane dataset				
100%	-	69.86	**70.52**	-	-	-
50%	-	69.39	68.59	69.68	**70.75**	70.41
10%	-	60.99	61.46	65.56	64.04	**66.10**
5%	-	56.61	57.78	61.99	62.32	**63.67**
1%	-	32.99	32.48	51.38	**55.10**	52.80

stantially from its origin usage. In general, semi-supervised models perform similar on the TuSimple dataset as it only includes the highway scenario. On the CULane dataset, ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ consistently outperforms ERFNet-$pseudo$, validating the usefulness of the Hough priors ($\mathcal{L}_{HT}$) in exploiting lane representations in the semi-supervised setting. The s4GAN is lacking since we are unable to produce reliable prediction.

We observe that ERFNet-HT-$\mathcal{L}_{HT}$ improves over all other models on the 1% subset by a large margin. On the 1% subset, there is not sufficient labelled data (less than $1K$ training images), and therefore the "ground truth" produced by pseudo-labelling in ERFNet-$pseudo$ is noisy and imperfect. In this case, learning from pseudo-labelled data explicitly can be harmful, while the $\mathcal{L}_{HT}$ avoids this problem by exploiting useful prior geometric knowledge about lines, in Hough space. In comparison, on the 50% subset, the differences among all models are marginal, when ample training data is available. The experiment demonstrates the potential of $\mathcal{L}_{HT}$ for data-efficient learning in a semi-supervised setting.

We compare the performance of all ERFNet models in various driving scenarios in Table 3.2. ERFNet-HT-$\mathcal{L}_{HT}$ shows considerable improvement over other models in most scenarios in Table 3.2, and the advantage accentuates (up to 5%), where the amount of labelled data is decreased to 1% only. The superiority of ERFNet-HT-$\mathcal{L}_{HT}$ demonstrates the capability of $\mathcal{L}_{HT}$ to exploit geometric lanes information from unlabelled data. We also notice that the "No line", "Shadow" and "Dazzle" scenarios are more challenging for all methods, compared with the other scenarios.

We visualize line predictions from different models in Figure 3.2. Our ERFNet-HT-$pseudo$+$\mathcal{L}_{HT}$ better localizes lanes, especially when a lane extends away from the image boundary, as in the first two examples. As shown in the second example, due to occlusion, ERFNet and ERFNet-HT miss the two middle lanes, while ERFNet-HT-$\mathcal{L}_{HT}$ only predicts

Table 3.2: F_1 **scores for different scenarios, with** 1% **labelled data.** ERFNet-HT-$\mathcal{L}_{HT}$ outperforms other models in most scenarios, indicating that the $\mathcal{L}_{HT}$ loss exploits useful geometric knowledge of lanes when adding unlabelled samples. (Note: for cross-road, we show only the number of false-positives, as in [10].)

ERFNet models	Baseline [5]	HT [9]	*pseudo*	HT-$\mathcal{L}_{HT}$	HT-*pseudo* +$\mathcal{L}_{HT}$
Normal	49.24	51.25	69.72	**75.06**	71.83
Crowded	31.74	31.49	49.53	**52.52**	50.97
Night	22.36	21.67	45.27	**50.77**	45.42
No line	18.78	17.54	28.05	**32.02**	30.12
Shadow	24.71	17.69	36.63	**38.50**	35.97
Arrow	39.39	38.22	57.28	**63.33**	59.18
Dazzle	26.25	23.76	**40.29**	40.28	39.42
Curve	33.62	34.53	46.56	**50.52**	46.42
Cross-road	6949	8711	**3355**	5292	3676
Avg F_1	33.00	32.48	51.38	**55.10**	52.80

Figure 3.2: **Visualizations of predicted lanes on the CULane dataset.** Only 10% annotated data is used for training. ERFNet-HT-*pseudo*+$\mathcal{L}_{HT}$ performs better on challenging samples and better localizes lane boundaries. The inference speed of the ERFNet-HT is around 13 frames per second on a NVIDIA GTX1080Ti GPU.

one. In the third example there is an annotation inconsistency, where the opposite lane at the image border is not annotated. Overall, ERFNet-HT-*pseudo*+$\mathcal{L}_{HT}$ produces sharper and more precise predictions, in both simple and challenging scenarios.

3.5. LIMITATIONS AND CONCLUSIONS

We propose semi-supervised lane detection by exploiting global line priors in Hough space through the use of an additional loss. We can incorporate unlabelled data during training thus overcoming the need for expensive and error-prone annotations. Currently our method assumes a single lane in each channel, and therefore we can optimize for the global maximum in Hough space. This assumption may not always hold and an extension to multiple local maxima is future research. However, our proposed Hough loss adds valuable prior geometric knowledge about lanes when annotations are too scarce even for pseudo-labelling based methods. We experimentally demonstrate the added value of our proposed loss on TuSimple and CULane datasets for limited annotated data.

REFERENCES

[1] B. He, R. Ai, Y. Yan, and X. Lang, *Accurate and robust lane detection based on dual-view convolutional neutral network,* in *Intelligent Vehicles Symposium* (2016) pp. 1041–1046.

[2] J. Pohl, W. Birk, and L. Westervall, *A driver-distraction-based lane-keeping assistance system,* Journal of Systems and Control Engineering **221**, 541 (2007).

[3] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, *Learning lightweight lane detection cnns by self attention distillation,* in *ICCV* (2019) pp. 1013–1021.

[4] Y. Hou, Z. Ma, C. Liu, T.-W. Hui, and C. C. Loy, *Inter-region affinity distillation for road marking segmentation,* in *CVPR* (2020) pp. 12486–12495.

[5] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, *Erfnet: Efficient residual factorized convnet for real-time semantic segmentation,* Transactions on Intelligent Transportation Systems **19**, 263 (2017).

[6] Z. Qin, H. Wang, and X. Li, *Ultra fast structure-aware deep lane detection,* in *The European Conference on Computer Vision (ECCV)* (2020).

[7] M. Huijser and J. C. van Gemert, *Active decision boundary annotation with deep generative models,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 5286–5295.

[8] R. O. Duda and P. E. Hart, *Use of the hough transformation to detect lines and curves in pictures,* Communications of the ACM **15**, 11 (1972).

[9] Y. Lin, S. L. Pintea, and J. C. van Gemert, *Deep hough-transform line priors,* European Conference on Computer Vision (2020).

[10] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, *Spatial as deep: Spatial cnn for traffic scene understanding,* CoRR (2017).

[11] *Tusimple lane detection challenge,* https://github.com/TuSimple/tusimple-benchmark.

[12] Z. Kim, *Robust lane detection and tracking in challenging scenarios,* Transactions on Intelligent Transportation Systems **9**, 16 (2008).

[13] H. Deusch, J. Wiest, S. Reuter, M. Szczot, M. Konrad, and K. Dietmayer, *A random finite set approach to multiple lane detection,* in *International Conference on Intelligent Transportation Systems* (2012) pp. 270–275.

[14] H. Yoo, U. Yang, and K. Sohn, *Gradient-enhancing conversion for illumination-robust lane detection,* Transactions on Intelligent Transportation Systems **14**, 1083 (2013).

[15] T. Wu and A. Ranganathan, *A practical system for road marking detection and recognition,* in *Intelligent Vehicles Symposium* (2012) pp. 25–30.

[16] B. Dorj and D. J. Lee, *A precise lane detection algorithm based on top view image transformation and least-square approaches,* Journal of Sensors **2016** (2016).

[17] K. Ghazali, R. Xiao, and J. Ma, *Road lane detection using h-maxima and improved hough transform,* in *International Conference on Computational Intelligence, Modelling and Simulation* (2012) pp. 205–208.

[18] Y. Wang, D. Shen, and E. K. Teoh, *Lane detection using spline model,* Pattern Recognition Letters **21**, 677 (2000).

[19] B. Yu and A. K. Jain, *Lane boundary detection using a multiresolution hough transform,* in *ICIP*, Vol. 2 (1997) pp. 748–751.

[20] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon, *Vpgnet: Vanishing point guided network for lane and road marking detection and recognition,* in *ICCV* (2017) pp. 1947–1955.

[21] J. Li, X. Mei, D. Prokhorov, and D. Tao, *Deep neural network for structural prediction and lane detection in traffic scene,* Transactions on neural networks and learning systems **28**, 690 (2016).

[22] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, *Towards end-to-end lane detection: an instance segmentation approach,* in *Intelligent vehicles symposium* (2018) pp. 286–291.

[23] J. Tang, S. Li, and P. Liu, *A review of lane detection methods based on deep learning,* Pattern Recognition , 107623 (2020).

[24] J. Zhang, Y. Xu, B. Ni, and Z. Duan, *Geometric constrained joint lane segmentation and lane boundary detection,* in *ECCV* (2018) pp. 486–502.

[25] J. E. Van Engelen and H. H. Hoos, *A survey on semi-supervised learning,* Machine Learning **109**, 373 (2020).

[26] A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy, *Semi-supervised learning using randomized mincuts,* in *ICML* (2004) p. 13.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations,* ICML (2020).

[28] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, *Semi-supervised learning with deep generative models,* NeurIPS **27**, 3581 (2014).

[29] A. Blum and S. Chawla, *Learning from labeled and unlabeled data using graph mincuts,* ICML (2001).

[30] T. Jebara, J. Wang, and S.-F. Chang, *Graph construction and b-matching for semi-supervised learning,* in *ICML* (2009) pp. 441–448.

[31] X. Zhu, Z. Ghahramani, and J. D. Lafferty, *Semi-supervised learning using gaussian fields and harmonic functions,* in *ICML* (2003) pp. 912–919.

[32] S. Mittal, M. Tatarchenko, and T. Brox, *Semi-supervised semantic segmentation with high-and low-level consistency,* IEEE transactions on pattern analysis and machine intelligence (2019).

[33] L. Bottou, *Large-scale machine learning with stochastic gradient descent,* in *Proceedings of COMPSTAT'2010* (2010) pp. 177–186.

4

DEEP VANISHING POINT DETECTION: GEOMETRIC PRIORS MAKE DATASET VARIATIONS VANISH

Deep learning has improved vanishing point detection in images. Yet, deep networks require expensive annotated datasets trained on costly hardware and do not generalize to even slightly different domains, and minor problem variants. Here, we address these issues by injecting deep vanishing point detection networks with prior knowledge. This prior knowledge no longer needs to be learned from data, saving valuable annotation efforts and compute, unlocking realistic few-sample scenarios, and reducing the impact of domain changes. Moreover, the interpretability of the priors allows to adapt deep networks to minor problem variations such as switching between Manhattan and non-Manhattan worlds. We seamlessly incorporate two geometric priors: (i) Hough Transform – mapping image pixels to straight lines, and (ii) Gaussian sphere – mapping lines to great circles whose intersections denote vanishing points. Experimentally, we ablate our choices and show comparable accuracy to existing models in the large-data setting. We validate our model's improved data efficiency, robustness to domain changes, adaptability to non-Manhattan settings.

This chapter is published as:
Yancong Lin, Ruben Wiersma, Silvia-Laura Pintea, Klaus Hildebrandt, Elmar Eisemann, and Jan van Gemert. *Deep vanishing point detection: Geometric priors make dataset variations vanish*. Conference on Computer Vision and Pattern Recognition (CVPR), 2022.

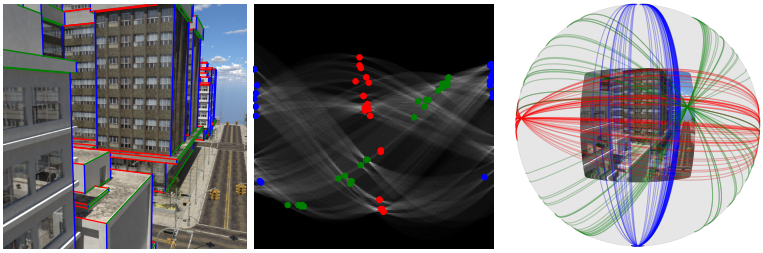


Image domain (i) Hough Transform (ii) Gaussian sphere

Figure 4.1: We add two geometric priors: (i) Hough Transform and (ii) Gaussian sphere mapping, for vanishing points detection. We transform learned image features to the Hough domain, where lines are mapped to individual bins. We further project the Hough bins to the Gaussian sphere, where lines become great circles and vanishing points are at the intersection of great circles. Each color represents a set of image lines related to a vanishing point. Adding geometric prior knowledge makes our model data-efficient, less dependent on domain-specifics, and easily adaptable to problem variations such as detecting a variable number of vanishing points.

4.1. INTRODUCTION

Vanishing point detection in images has non-vanishing real-world returns: camera calibration [1–3], scene understanding [4], visual SLAM [5, 6], or even autonomous driving [7]. Deep learning is an excellent approach to vanishing point detection [8–11], where all geometric knowledge is learned from large annotated data sets. Yet, in the real-world, there are several factors that complicate deep learning solutions: (1) Manually annotating large training sets is expensive and error prone; (2) Training models on large data sets require costly computational resources; (3) Practical changes to data collection cause domain shifts, hampering deep network generalization; (4) Slight changes in the problem setting require a complete change in deep network architectures. Thus, there is a need to make deep learning less reliant on data, and its architectures more robust to variants of the same problem.

In this paper, we add geometric priors to deep vanishing point detection. Using geometric priors is data-efficient as this knowledge no longer needs to be learned from data. Thus, fewer annotations and compute resources are needed. Moreover, by relying on priors, the model is less sensitive to particular idiosyncrasies in the training data and generalizes better to domains with slightly different data distributions. Another advantage of a knowledge-based approach is that it is interpretable, and thus the architecture is easy to adapt to a slightly different problem formulation.

We add two geometric priors, see Figure 4.1: (i) the Hough Transform and (ii) a Gaussian sphere mapping. Our trainable Hough Transform module represents each line as an (offset, angle) pair in line polar coordinates, allowing us to identify individual lines in Hough space [12]. We subsequently map these lines from Hough space to the Gaussian sphere, where lines become great circles, and vanishing points are located at the intersection of great circles [13]. The benefit of using great circles is that lines are mapped from the unbounded image plane to a bounded unit sphere, facilitating vanishing point detection outside the image view. Both the Hough Transform and the Gaussian sphere mapping are

end-to-end trainable, taking advantage of learned representations, while adding knowledge priors.

This paper makes the following contributions: (1) we add two geometric priors for vanishing point detection by mapping CNN features to the Hough Transform, and mapping Hough bins to the Gaussian sphere; (2) we validate our choices and demonstrate similar accuracy as existing models on the large ScanNet [14] and SceneCity Urban 3D [15]; (3) we show that adding prior knowledge increases data-efficiency, improving accuracy for smaller datasets; (4) we demonstrate our ability to tackle a different problem variant: detecting a varying numbers of vanishing points on the NYU Depth [16] dataset, where the number of vanishing points varies drastically from 1 to 8; (5) we show that adding prior knowledge reduces domain shift sensitivity, which we validate by cross-dataset testing.

4.2. RELATED WORK

Geometry-based vanishing point detection. Vanishing points occur at intersections of straight lines. Lines can be found by contour detection [17] or a dual point-to-line mapping [18]. The common approach, however, is using an explicit straight-line parameterization in the Hough Transform [19–21]. We exploit this straight-line parameterization as prior knowledge in a Hough Transform module.

Combining lines to vanishing points can be done by measuring the probability of a group of lines passing through the same point [22], voting schemes [23, 24], or hypothesis testing by counting the number of inlier lines such as J-Linkage [25] used in [26, 27]. Other approaches see vanishing point detection as a grouping problem by applying line clustering [28–30], expectation-maximization [2, 31–33], or branch-and-bound [34–36]. While these methods work well, they do not exploit prior knowledge of the 3D world.

A strong geometric prior for vanishing points is modeled by the Gaussian-sphere [13, 37]. A line in an image represents a great circle on the Gaussian-sphere and the intersections of great circles on the sphere denote vanishing points, detected as local maxima [13, 20]. Mapping lines to the Gaussian sphere shifts the problem from the unbounded image plane to the constrained parameter space defined by the Gaussian sphere [37–39]. Constraining the search space is a form of regularization, and is particularly beneficial for limited-data deep learning. We exploit this prior knowledge by incorporating the Gaussian sphere mapping.

Learning-based vanishing point detection. Vanishing point detection can be learned from large annotated datasets [8–10, 40]. It is effective to split the problem in separate stages: line detection, inverse gnomonic projection, network training and post-processing, as in Kluger *et al.* [41]. Conic convolutions on hemisphere points, provided further improvements, in Zhou *et al.* [11]. In contrast, rather than focusing on accurate large-scale deep models, we consider challenging real-world scenarios, such as: limited training samples, cross-dataset domain switch, and non-Manhattan world.

Robustness to domain shifts. Classical solutions to vanishing point detection [26, 42, 43] are built exclusively on prior-knowledge. Such methods are data-free, and thus designed to work on any domain. Yet, they cannot take advantage of expressive deep-feature learning for vanishing point detection [11, 44]. On the other hand, deep models are notoriously sensitive to distribution shifts between training and test [45, 46]. Active research on this includes: domain adaptation [47–49], domain generalization [50], multi-domain learning [51, 52],

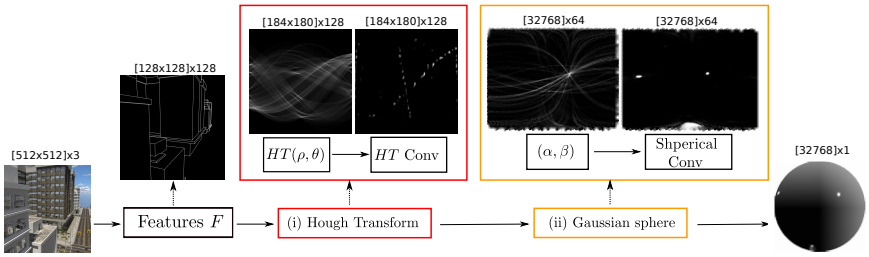


Figure 4.2: **Overview:** The model starts from in input image, and predicts vanishing points on the Gaussian hemisphere by relying on two geometric priors: (i) Hough Transform, and (ii) Gaussian sphere mapping. We use a convolutional network to learn features which are then mapped to Hough space, where each bin is a line. We filter the Hough space and project Hough bins to the Gaussian hemisphere and apply spherical convolutions to find vanishing points. We indicate the size of the learned features above, where the last dimension is the number of channels. We sample 32,768 points on the hemisphere using the Fibonacci lattice [59], resulting in features maps of size 32,768. Our model learns to classify spherical points as vanishing points or not using a binary cross-entropy loss. There is *no* intermediate supervision.

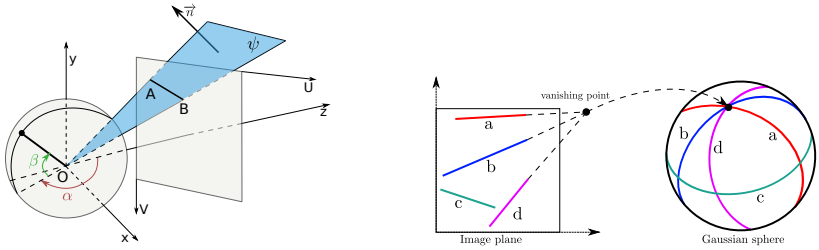
etc. Such solutions entail significant changes to the deep network model, adding complexity for practical real-world applications. Hence, we focus on a single method which does not requiring large model changes for robustness to minor domain shifts. Our goal is to combine the robustness of knowledge-based methods, with the power of deep representation learning.

Manhattan versus non-Manhattan world. The Manhattan world assumes exactly 3 vanishing points. This assumption has been proven useful for orthogonal vanishing point detection [34, 53–55]. However, the Manhattan assumption does not hold in several real-world scenarios such as non-orthogonal walls and wireframes in man-made structures. Vanishing point detection in non-Manhattan world is done by robust multi-model fitting [56], horizon line detection [40], branch-and-bound with a novel mine-and-stab strategy [57], Bingham mixture model fitting [58] or non-maximum suppression on the Gaussian sphere [44]. In our work, we refrain from adding explicit orthogonality constraints, which makes our method applicable to non-Manhattan scenarios as well. We rely on the Hough Transform and the Gaussian sphere to map pixel-wise representations to the entire hemisphere. And using a clustering algorithm we detect multiple vanishing points simultaneously.

4.3. GEOMETRIC PRIORS FOR VP DETECTION

General outline of our approach. Figure 5.2 depicts the overall structure of our model. We build on two geometric priors: (i) Hough Transform, and (ii) Gaussian sphere mapping. A CNN learns image features, which are then mapped to a line parameterization via Hough Transform. We project the features of parameterized lines to the Gaussian sphere where spherical convolutions precisely localize vanishing points.

(i) Hough Transform. Similar to [11], we use a single-stack hourglass network [60] to extract image features, F to be mapped into Hough space [61], HT. The HT space parameterizes image lines in polar coordinates using a set of discrete offsets ρ and discrete



(a) The Gaussian sphere (b) Vanishing points on the Gaussian sphere

Figure 4.3: **Gaussian sphere representations for vanishing points [13, 20]:** (a) The Gaussian sphere is a unit sphere located at the camera center, **O**. Points on the sphere are encoded by two angles: (α, β) the azimuth and the elevation, respectively. A line segment **AB** in the image plane together with the camera center **O** forms a plane ψ, highlighted in blue. To define the mapping from the image to the sphere, we only need to know the normal $\vec{n}$ to the plane ψ. (b) Image lines are projected as great circles on the sphere. The intersection of multiple great circles on the sphere represents a vanishing point.

angles θ, defining a 2D discrete histogram. In practice, a set of pixels $(x(i), y(i))$ along a line indexed by i, vote for a line parameterization to which they all belong:

$$HT(\rho,\theta) = \sum_i F(\rho\cos\theta - i\sin\theta, \rho\sin\theta + i\cos\theta) \tag{4.1}$$

The Hough Transform module starts from an $[H \times W]$ feature map F and outputs an $[N_\rho \times N_\theta]$ Hough histogram HT, where N_ρ and N_θ are the number of sampled offsets and angles in Hough Transform. We set H=128, W=128, N_ρ=184, and N_θ=180. This results in $[N_\rho \times N_\theta]$ possible line parameterizations. We find the local maxima in the Hough domain by performing a 1D convolutions over the offsets. This removes the noisy responses in the Hough space, as in Figure 5.2. We refer the readers to [61] for details.

(ii.1) Gaussian sphere mapping. The Gaussian sphere is a unit sphere centered at the camera origin, **O**. Vanishing points on the sphere are represented as normalized 3D line directions δ.

Starting from a bin in the Hough domain (ρ_{AB}, θ_{AB}), corresponding to a line direction in the image plane $\overrightarrow{AB}$, we want to map this to the Gaussian sphere. Two image points **A** and **B** sampled from a line represented by its HT bin (ρ_{AB}, θ_{AB}), together with the camera center **O**, form a plane ψ as depicted in Figure 4.3(a). The plane ψ is described by its normal vector:

$$\vec{n} = (n_x, n_y, n_z) = \frac{\overrightarrow{OA} \times \overrightarrow{OB}}{\left\| \overrightarrow{OA} \times \overrightarrow{OB} \right\|}. \tag{4.2}$$

This normal vector $\vec{n}$ is the only information we need to map the image line direction $\overrightarrow{AB}$ to the Gaussian sphere.

The spherical coordinates (α, β) describe a point on the Gaussian sphere, where α is the azimuth defined as the angle from the z-axis in the xz plane, and β is the elevation representing the angle measured from the xz plane towards the y-axis, as shown in Figure 4.3(a).

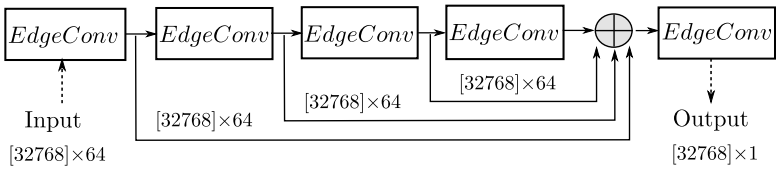


Figure 4.4: **Spherical convolutions on the hemisphere.** We use EdgeConv [62] for precise vanishing point localization on the Gaussian sphere. The concatenation of the previous feature maps is fed into the final layer to produce a prediction.

The intersection between the plane ψ and the Gaussian sphere is a great circle. This great circle represents the projection of the image line direction $\overrightarrow{AB}$ on the Gaussian sphere. Intersections of multiple great circles are potential vanishing points, see Figure 4.3(b).

We compute the projection of the image line direction $\overrightarrow{AB}$, by estimating the elevation β as a function of the azimuth α and the normal vector $\vec{n}$ [20]:

$$\beta(\alpha, \vec{n}) = \tan^{-1} \frac{-n_x \sin\alpha - n_z \cos\alpha}{n_y}, \tag{4.3}$$

where we uniformly sample α in the range $[-\pi/2, \pi)$.

Because the Gaussian sphere is symmetric we only need a hemisphere. We sample N points on the Gaussian hemisphere using a Fibonacci lattice [59] and then project lines, corresponding to bins in the Hough space, to these N sampled sphere points. For each line parameterization in Hough space (ρ, θ), we first compute its normal vector $\vec{n}$. We, then, estimate its corresponding (α, β) spherical coordinates using Eq. (4.3). We subsequently assign each (α, β) pair to its nearest neighbor in the sampled points from the Fibonacci lattice, by computing their cosine distance. To parallelize this process, we precompute the projection of all Hough line parameterizations onto the sampled sphere locations. This mapping is stored in an $[N_\rho \times N_\theta \times M]$ tensor, where $[N_\rho \times N_\theta]$ is the number of line parameterizations in Hough space and M is the number of sampled azimuth angles α. We set N=32,768 and M=1,024.

(ii.2) Spherical convolutions on the hemisphere. We employ spherical convolutions to predict vanishing points. We treat the points sampled on the hemisphere as a point cloud and use EdgeConv [62] to convolve over the hemisphere. EdgeConv operates on a k-nearest neighbor graph on the points. It learns to represent local neighborhoods by applying a non-linear function to the neighbors' features, and then aggregates those features with a symmetric operator. The neighbors' features are localized by subtracting the features of the centroid. Like [62], we take the per-feature maximum over the neighbors to aggregate edge features.

As shown in Figure 4.4, the spherical part of our model contains 5 EdgeConv modules [62]. Each EdgeConv module transforms neighboring features with a fully connected layer, a BatchNorm layer [63] and a LeakyReLU activation. We use N=32,768 nodes on the hemisphere and compute the 16 nearest neighbors for each node. We concatenate the features maps from previous layers and feed them into the last EdgeConv layer to produce the final prediction.

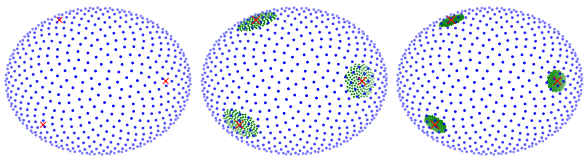


Figure 4.5: **Multi-scale sampling on the hemisphere.** We sample points at three scales for detecting vanishing points in the Manhattan world, as in [11]. Blue indicates sampling at the first scale, while green indicates fine-grained sampling at the following scales. The red crosses are the predictions at each scale.

Model training and inference. We train the model using the binary cross-entropy loss. For each annotated vanishing point, we label its nearest neighbor in the sampled points as +1 and the others as 0. Because the number of positive samples is considerably lower than the negative samples, we compute two separate average losses over the positive and the negative samples, and then sum these. There is *no* intermediate supervision or guidance.

During inference, we use DBSCAN [64, 65] to cluster all points on the Gaussian sphere based on the cosine distance. The *eps* parameter of DBSCAN [65] is set to be 0.005. The point with the highest confidence in each cluster is the prediction. We rank all predictions by confidence.

Multi-scale sampling with the Manhattan assumption. For the Manhattan world, we know beforehand that there are only 3 orthogonal vanishing points, therefore in this scenario, we can use a multi-scale sampling strategy to reduce computation, as in [11]. Here, we sample points and apply spherical convolutions at 3 scales: $\delta \approx \{90^\circ, 13^\circ, 4^\circ\}$ and $N = \{512, 128, 128\}$, where δ controls the sampling radius and N indicates the number of sampled points respectively. Figure 4.5 displays the multi-scale sampling. The spherical convolution networks share the same architecture while processing different number of samples. We provide details in the supplementary material.

4.4. EXPERIMENTS

Datasets. We evaluate on three datasets following the Manhattan world assumption: SU3 (SceneCity Urban 3D) [15], ScanNet [14], YUD [31], as well as the NYU Depth [16] dataset which does not follow the Manhattan world assumption. The SU3 dataset contains 23K synthetic images, which are split into 80%, 10% and 10% for training, validation and testing respectively. The ScanNet has more than 200K real-world images, among which 189,916 examples are used for training. The "ground truth" VPs are estimated from surface normals as in [11], thus being less precise than other datasets. In the NYU Depth dataset, the number of vanishing points varies from 1 to 8 across images, making it more challenging. The NYU Depth dataset has 1,449 images, approximately ×200 and ×20 smaller than the ScanNet dataset and the SU3 dataset, respectively, further increasing the difficulty of training CNN models. We additionally demonstrate the effect of geometric priors on the small-scale YUD dataset with only 102 images. Detailed comparisons are in the supplementary material. Unless specified otherwise, we use the ground-truth focal length on SU3, ScanNet and YUD for the Manhattan assumption.

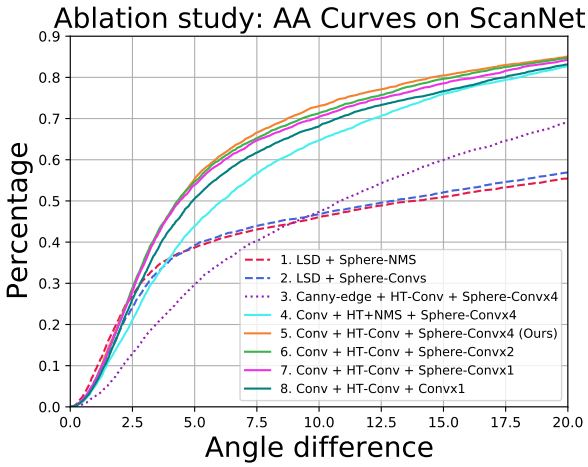


Figure 4.6: **Exp 1: Model choices.** We show the effects of the two geometric priors quantitatively on the ScanNet-1% subset. Adding HT layers and spherical convolutions outperforms the baselines, thus demonstrating the effectiveness of geometric priors.

	HT (# of angles)		Sphere (# of points)		
	90	180	$8K$	$16K$	$32K$
AA@3°	77.1	79.3	73.3	77.4	79.3

Table 4.1: **Quantization analysis on SU3-10% subset.** Denser samplings improve performance. In practice, we uniformly sample 180 angles from $[0, \pi)$ for HT, and $32K$ points on the sphere.

Evaluation. On the SU3, ScanNet and YUD datasets (Manhattan assumption), we evaluate the angle difference between the predicted and the ground-truth vanishing points in the camera space, as in [11, 44, 56]. We then estimate the percentage of the predictions that have a smaller angle difference than a given threshold and compare the angle accuracy (AA) under different thresholds, as in [11, 44]. We use the ground-truth focal length to exploit the orthogonal constraint. On the NYU Depth dataset we follow [56] and first rank detected vanishing points by confidence, and then use the bipartite matching [66] to calculate the angular errors for the top k predictions. After matching, we generate the recall curve and measure the area under the curve (AUC) up to a threshold, e.g. 10°.

Baselines. We compare our model with J-Linkage [26], Contrario-VP [43], Quasi-VP [42, 67], NeurVPS [11] and CONSAC [56] on SU3, ScanNet and YUD. On the non-Manhattan NYU Depth dataset we only compare with J-Linkage, T-Linkage [68], CONSAC and VaPid [44], as the other models rely on the Manhattan assumption. J/T-Linkage, Contrario-VP and Quasi-VP are non-learning methods, employing line segment detection [69]. NeurVPS and our model are end-to-end trainable, while CONSAC needs line segments as inputs. We follow the official implementations and use the default hyperparameters to reproduce all results. We do not consider the baselines [24, 44, 58] due the lack of code/results on certain datasets.

Datasets			SU3 [15]		ScanNet [14]			YUD [31]		
Metrics	Params	FPS	AA@3°	AA@5°	AA@3°	AA@5°	AA@10°	AA@3°	AA@5°	AA@10°
J-Linkage† [26]	—	1.0	82.0	87.2	15.7	27.3	43.0	60.8	71.8	81.5
Contrario-VP† [43]	—	0.6	64.8	72.2	12.0	21.4	35.3	58.6	70.7	81.8
Quasi-VP [67]	—	**29.0**	75.9	80.7	14.7	25.3	39.4	58.6	61.0	74.0
CONSAC† [56]	0.2 M	3.0	86.3	90.3	15.8	24.6	36.0	**61.7**	73.6	84.4
NeurVPS [11]	22 M	0.5	**93.9**	**96.3**	24.0	41.8	**64.4**	52.4	64.0	77.8
Ours	7 M	5.5	84.0	90.2	**24.8**	**42.1**	63.7	60.7	**74.3**	**86.3**
*Ours**	5 M	23.0	84.8	90.7	22.9	39.8	62.4	59.5	72.6	85.4
Ours†	7 M	5.5	81.7	88.7	22.2	38.8	59.9	59.1	72.6	84.6

Table 4.2: **Exp 2: Manhattan world.** Angular accuracy on SU3, ScanNet and YUD datasets. *Ours* achieves the best results on the the YUD dataset, and is competitive on the larger ScanNet and SU3 datasets. *Ours** adopts the multi-scale sampling strategy, thus being significantly faster. † assumes unknown focal length, thus making the Manhattan assumption no longer applicable. *Ours*† shows a constant decrease over *Ours* across datasets, indicating the usefulness of the orthogonal constraint. Supplementary material provides qualitative visualizations. We conclude that adding priors does not reduce accuracy in the large scale setting.

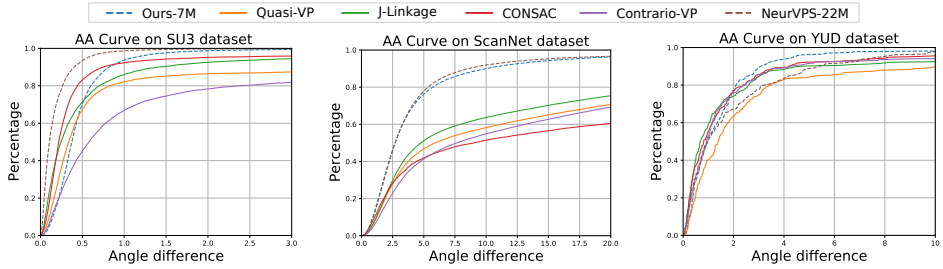
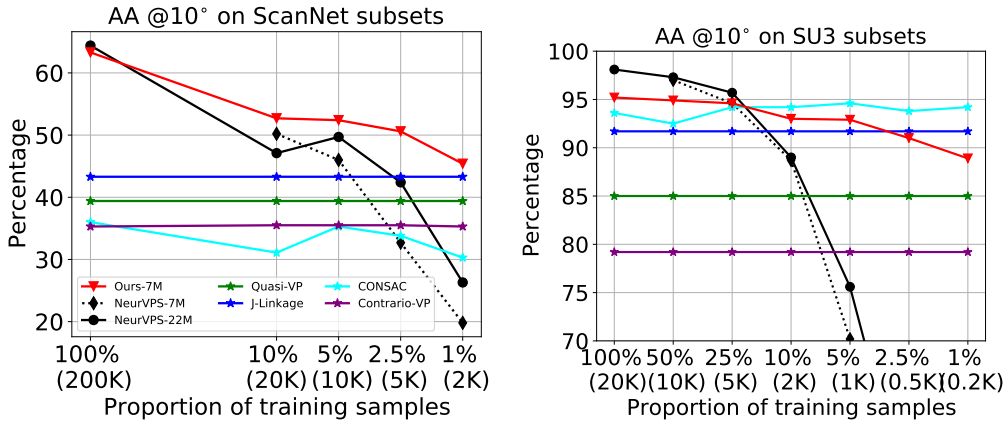


Figure 4.7: **Exp 2: Manhattan world.** AA curves on on the ScanNet, SU3 and YUD datasets containing 3 orthogonal vanishing points. Learning-based approaches outperform methods relying purely on line segments and grouping, validating the power of representation learning. Our model shows comparable results to the best performing NeurVPS on ScanNet, while using 3× less parameters. On the smaller YUD dataset, our model slightly exceeds state-of-the-art. Generally, with ample data, our approach is comparable to others.

Implementation details. We implement our model in Pytorch [70], and provide the code online [1]. Our models are trained from scratch on Nvidia RTX2080Ti GPUs with the Adam optimizer [71]. The learning rate and weight decay are set to be 4×10^{-4} and 1×10^{-5}, respectively. To maximize GPU usage, we set the batch size to 4 and 16 when using multi-scale sampling. On the SU3 and NYU Depth datasets, we train the model for a maximum of 36 epochs, with the learning rate decreases by 10 after 24 training epochs. On the ScanNet dataset, we train for 10 epochs and decay the learning rate by 10 after 4 epochs. On the YUD datset we use pre-trained models on SU3.

[1] https://github.com/yanconglin/VanishingPoint_HoughTransform_GaussianSphere



(a) Data efficiency on ScanNet (b) Data efficiency on SU3

Figure 4.8: **Exp 3.(a): Reduced data.** We report AA @10° on various subsets of the ScanNet and SU3 datasets, and indicate the number of parameters in the legend. On the ScanNet dataset, we outperform other methods on the 10%, 5% and 2.5% subsets. Our model degrades gracefully when reducing the training samples from 20K to 1K on the SU3 subset, while NeurVPS has a drastic drop in accuracy. CONSAC achieves top results on SU3 due to pre-extracted line segments, but fails on ScanNet because of inaccurate line detection. There is a similar trend for the baselines relying on line segment detection. Our model predictions are stable having small variances (±0.50 and ±0.43 on the 1% subsets of ScanNet and SU3 respectively) across 3 repetitions. This experiment validates the data efficiency of our model.

4.4.1. EXP 1: EVALUATING MODEL CHOICES

We evaluate on a subset of ScanNet containing 1% of the data, and provide the results in Figure 4.6. Model *(1)* is a non-learning baseline using a classic line segment detector (LSD) [69] and non-maximum suppression (NMS) on the sphere. Model *(2)* replaces the NMS with spherical convolutions, but still shows inferior result as LSD fails to detect reliable line segments. Model *(3)* combines a Canny-edge detector, Hough Transform and spherical convolutions. Comparing *(3-5)* indicates the added value of learning semantics from images, rather than using classic edge detectors. Comparing *(4-5)* shows the effectiveness of backpropagating through Hough Transform. Comparing *(5-8)* exemplifies the added value of spherical convolutions. Our method combines both classical and deep learning approaches into an end-to-end trainable model.

We also evaluate the impact of quantizations numerically on the synthetic SU3-10% subset, which contains precise VP annotations, thus making quantization a crucial factor. As shown in Table 4.1, fine-grained sampling is essential for a better result.

4.4.2. EXP 2: VALIDATION ON LARGE DATASETS

We validate that adding prior knowledge does not deteriorate accuracy when there is plenty of data. We compare to five state-of-the-art baselines [11, 26, 43, 56, 67] on the ScanNet, SU3 and YUD datasets. On the ScanNet and SU3 datasets, we train all learning models from scratch on the full training split. On the YUD dataset, we use the pre-trained models

on SU3 without fine-tuning. For CONSAC and J-Linkage, we select top-3 predictions. We also measure the inference speed on a single RTX2080 GPU. Multi-scale sampling *Ours** achieves 23 FPS, a large speedup over the vanilla design, as we utilize the orthogonality for efficient sampling.

Table 4.2 shows the AA scores on the ScanNet, SU3 and YUD datasets, while Figure 4.7 depicts AA curves for varying angle differences. The SU3 dataset is easier as most images contain strong geometric cues (e.g. sharp edges and contours); this is no longer the case in the ScanNet dataset. The prediction error on the more realistic ScanNet dataset is significantly larger for all methods. On the ScanNet dataset, NeurVPS and our model are visibly better than methods relying on predefined line segments as inputs. The main advantage of NeurVPS and our model is their ability to learn useful feature representations directly from images. On the SU3 dataset, NeurVPS exceeds the other methods in the low-error region (from 0° to 1°). J-Linkage, Quasi-VP and CONSAC have similar results, and all of them stabilize at 1°. On SU3 our model is less accurate in 0°-1°, yet it compensates at ≥ 1°. Our inferior performance in 0°-1° range results from the quantization errors in Hough Transform and the Gaussian sphere mapping. On the small-scale YUD dataset [31], our model achieves comparable accuracy without fine-tuning, and exceeds the other methods in the ≥ 2° area, indicating the generalization ability of our model in the small data regime. We conclude that our model using prior knowledge performs similar to existing solutions.

4.4.3. EXP 3: CHALLENGING SCENARIOS

EXP 3.(A): REDUCED DATA

We evaluate data efficiency by reducing the number of training samples to {10%, 5%, 2.5%, 1%} on the ScanNet dataset, resulting in approximately 20K, 10K, 5K and 2K training images. Similarly, we also sample the SU3 dataset into {50%, 25%, 10%, 5%, 2.5%, 1%} subsets. We train all learning models from scratch using the default hyperparameters on each subset.

In Figure 4.8 we compare the AA scores at 10° with state-of-the-art methods. We use our vanilla design without the multi-scale sampling speedup. The first thing to notice is that non-learning methods are robust to data reduction. Yet, non-learning methods cannot take any training data into account, and thus they do not perform as well when more data is available, as we validated in the previous experiment. On the ScanNet dataset, our model visibly exceeds the other methods on the 10%, 5% and 2.5% subsets. In comparison, NeurVPS suffers from large accuracy decreases on small training data subsets. When decreasing the number of samples to 2K (1% subset), we still achieve competitive accuracy when compared to the non-learning methods, while NeurVPS fails to make reasonable predictions due to the lack of data. This shows the capability of our model to learn from limited data, thanks to the added geometric priors.

The NeurVPS model has ×3 more parameters than our model due to its fully-connected layer with 16M parameters. For fairness, we also consider 'NeurVPS-7M' with reduced fully-connected layers, having a similar number of parameters with our model. Both NeurVPS variants perform similar on various subsets. On the SU3 dataset the accuracy of NeurVPS decreases significantly when reducing the training dataset size, despite its superiority on the large training subsets. In comparison, our model degrades gracefully when training data decreases from 20K to 1K. Notably, on the 1% subset, with only 200 images for training, we are still able to achieve comparable performance with non-learning methods.

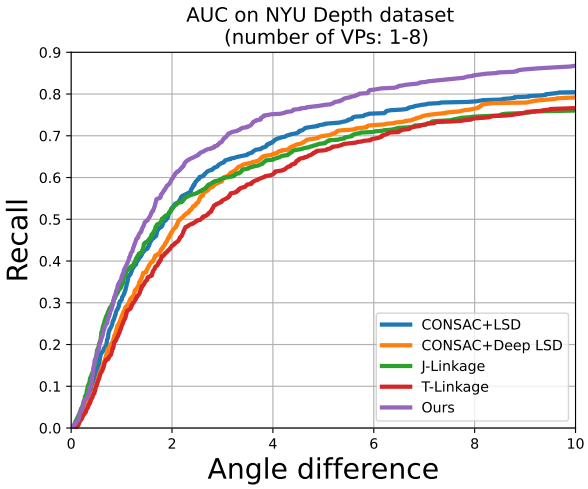


Figure 4.9: **Exp 3.(b): Non-Manhattan scenario.** We plot the recall curve on the non-Manhattan NYU dataset. Our method outperforms state-of-the-art, illustrating the ability of the model to detect a varying number of vanishing points. See supplementary material for qualitative visualizations.

Datasets	NYU Depth [16]			
	top-k = #gt		top-k = #pred	
AUC	@5°	@10°	@5°	@10°
J-Linkage[26]	49.30	61.28	54.48	68.34
T-Linkage[68]	43.38	58.05	47.48	64.59
CONSAC[56]	49.46	65.00	54.37	69.89
CONSAC[56]+ DLSD[61]	46.78	61.06	49.94	65.96
VaPiD [44]	-	69.10	-	-
Ours	**55.92**	**69.57**	**57.19**	**71.62**

Table 4.3: **Exp 3(b): Non-Manhattan scenario.** We report AUC scores on the NYU Depth dataset. Here "top-k = #gt" indicates the k most confident predictions where k is the number of annotated instances [56], while for "top-k =#pred" all predictions are used for evaluation. Our model exceeds state-of-the-art when detecting a varying number of vanishing points.

EXP 3.(B): NON-MANHATTAN SCENARIO

We compare with state-of-art methods in a more realistic non-Manhattan scenario with limited annotated data. CONSAC [56] uses the line segment detection in [69]. We also consider a variant of CONSAC with the more recent line segment detector in [61].

Figure 4.9 and Table 4.3 display the recall curve and the AUC values on the NYU Depth dataset, respectively. Our model consistently outperforms state-of-the-art baselines, and the improvement is more pronounced for larger angular differences. Although achieving the second-best result, VaPiD [44] assumes a constant number of instances and requires

Synthetic - real-world data						
Train	SU3 [15]					
Test	ScanNet [14]			YUD [31]		
Models	*Ours*	NeurVPS	CONSAC	*Ours*	NeurVPS	CONSAC
AA@3°	**15.2**	11.1	10.1	60.7	53.8	**61.7**
AA@5°	**25.9**	20.3	17.3	**74.3**	65.6	73.6
AA@10°	**39.5**	35.5	27.2	**86.3**	79.7	84.4
Real-world data						
Train	NYU Depth [16]					
Test	ScanNet [14] (AA)		YUD [31] (AA)		YUD+ [31] (AUC)	
Models	*Ours*	CONSAC	*Ours*	CONSAC	*Ours*	CONSAC
@10°	**33.6**	30.3	**83.2**	82.7	71.4	**75.0**

Table 4.4: **Exp 3.(c) Cross-dataset domain switch.** "Train" and "Test" specify the training and test datasets. CONSAC uses pre-extracted lines, thus being accurate on YUD/YUD+. However, its accuracy is lower on ScanNet due to the lack of reliable lines. In comparison, *Ours* is more accurate on both ScanNet and YUD without tuning. Our geometric priors improve the transferability of the model across datasets.

non-maximum suppression, which often results in over- and under-prediction. Our model outperforms existing methods by exploiting geometric priors, while not limiting the number of vanishing points detected.

EXP 3.(C): CROSS-DATASET DOMAIN SWITCH

We conduct cross-dataset test on multiple datasets, as displayed in Table 4.4. We compare with NeurVPS and CONSAC, which achieve top accuracy on individual datasets. When generalizing from synthetic dataset to the real-world (e.g. from SU3 to YUD), our model shows comparative results to CONSAC, which relies on prior line segment detection, making it robust to domain shifts. We observe a similar trend on real-world datasets (e.g. from NYU to YUD). However, on the challenging ScanNet dataset, *Ours* exceeds CONSAC, indicating the advantage of learning semantics over using pre-extracted lines. In contrast, NeurVPS does not transfer well to another dataset. This validates the robustness of the two priors in tackling domain shifts.

4.5. CONCLUSIONS AND LIMITATIONS

This paper focuses on vanishing point detection relying on well-founded geometric priors. We add two geometric priors as building blocks in the deep neural networks for vanishing point detection: Hough Transform and Gaussian sphere mapping. We validate experimentally the added value of our geometric priors when compared to state-of-the-art Manhattan methods, and show their usefulness on realistic/challenging scenarios: with reduced sam-

ples, in the non-Manhattan world where the challenge is to predict a varying number of vanishing points without the orthogonality assumption, and across dataset domains.

Despite of these improvements, our model also has several limitations. We pre-compute offline the mapping from images to the Hough bins and to the Gaussian sphere by fixing the size of the Hough histogram, as well as the Fibonacci sampling. However, these samplings introduce quantization errors which set an upper bound on accuracy. This is the primary reason for the limited accuracy on the SU3 dataset in the low-error region. A future research avenue is exploring an analytical mapping from image pixels to the Gaussian sphere. In addition, our model still relies on hundreds of fully labeled samples for training. One might consider testing the added priors in an unsupervised or weakly-supervised manner.

4.6. SUPPLEMENTARY MATERIAL

4.6.1. MULTI-SCALE SAMPLING ON THE GAUSSIAN SPHERE

Inspired by [11], we use a multi-scale sampling strategy to detect three orthogonal vanishing points in the Manhattan world. We start by uniformly sampling $N_{s=0}$ points at scale $s = 0$ on the entire hemisphere. We input these points into a spherical convolution network. Sequentially, we use the Manhattan assumption to choose 3 orthogonal vanishing points with the highest confidence as anchors. We uniformly sample $N_{s=1}$ points around each anchor in a local neighborhood defined by the radius $\delta_{s=1}$ at scale $s = 1$. Then, we feed these newly sampled points into a spherical convolution network. Finally, the point with highest confidence in each local neighborhood is considered as the anchor for sampling at the $(s+1)$th scale. Specifically, we set $\delta \approx \{90°, 13°, 4°\}$ and $N = \{512, 128, 128\}$. The spherical convolution networks share the same architecture while processing different number of samples. During training, we assign the nearest neighbors to the ground truth as positive samples while the others are considered as negative samples. We compute the cross-entropy losses averaged over positives and negatives respectively, at each scale.

4.6.2. DATASETS

Datasets	Images	Manhattan	Size	VPs	Train	Valid	Test
SU3 [2][15]	Synthetic	✓	512 × 512	3	18400	2300	2300
ScanNet [14]	Real	✓	512 × 512	3	189916	500	20942
YUD [31]	Real	✓	480 × 640	3	25	-	77
NYU [16]	Real	×	480 × 640	1-8	1000	224	225

Table 4.5: **Comparison of the four datasets.** The SU3, ScanNet and YUD datasets follow the Manhattan assumption with 3 orthogonal vanishing points, while the NYU Depth dataset is annotated with a varying number of instances. In addition, the size of the four datasets varies substantially. There are only 1000 and 25 training images in NYU and YUD datasets.

Table 4.5 shows a detailed comparison among all datasets, and Figure 4.10 displays image examples from each dataset. The SU3 dataset is synthetic and all images are well-

[2]Since there are no official splits, we divide all images into 80%/10%/10% partitions for training, validation and testing, respectively.

Figure 4.10: **Examples from the SU3, ScanNet, YUD and NYU Depth (labeled with ground truth lines) datasets.** Images from the SU3 dataset are well-calibrated with clear geometric cues, such as sharp edges and contours. In contrast, the other datasets capture real-world images where image content varies significantly. The NYU Depth dataset is labeled with multiple vanishing points (varying from 1-8).

calibrated with sharp edges. The ScanNet dataset captures indoor scenes in the real-world environments, where image content varies significantly. The YUD dataset captures both indoor and outdoor scenes in urban cities and contains only 102 images. SU3, ScanNet and YUD datasets follow the Manhattan world assumption where there are 3 orthogonal vanishing points. In comparison, the NYU Depth dataset has a varying number of instances across images. Moreover, there are 1449 images in total, and therefore training deep networks is highly challenging on the NYU Depth dataset due to the lack of data.

4.6.3. VISUALIZATIONS

We visualize predictions on the NYU Depth dataset in Figure 4.11. We show the input images, labeled line segments and detected vanishing points on the hemisphere. Each color represents a group of lines and their corresponding vanishing point. In the third row, our model correctly detects all vanishing points, as the colored × and ∘ overlap. In comparison, CONSAC fails to localize the red one and J-Linkage is unable to detect the green one. In addition, CONSAC makes nearby predictions: e.g., the blue and pink × markers in second row. This is caused by the LSD [69] method producing a large number of outlier segments, resulting in incorrect predictions. Our method is suitable for real-world scenarios, where the image content varies substantially.

Figure 4.12 and Figure 4.13 show detected vanishing points from our model on the SU3 and YUD datasets, respectively. Since all methods make reasonably good prediction and the difference is hardly visible, we only visualize our results.

Figure 4.14 compares detected vanishing points from all models on the ScanNet dataset. We compare all models in a column-wise manner, where the input image is on the top, while predictions from each method is displayed sequentially. We show the top 3 vanishing points for J-Linkage [26] and CONSAC [56]. In general, NeurVPS and ours are able to localize vanishing points more precisely than other non-learning approaches. As shown in the fourth example where the object is not orthogonally placed, Quasi-VP [67] fails due to the presence of strong outliers and the lack of inliers. This shows the disadvantage of non-learning method in dealing with complicated real-world scenarios. J-Linkage and

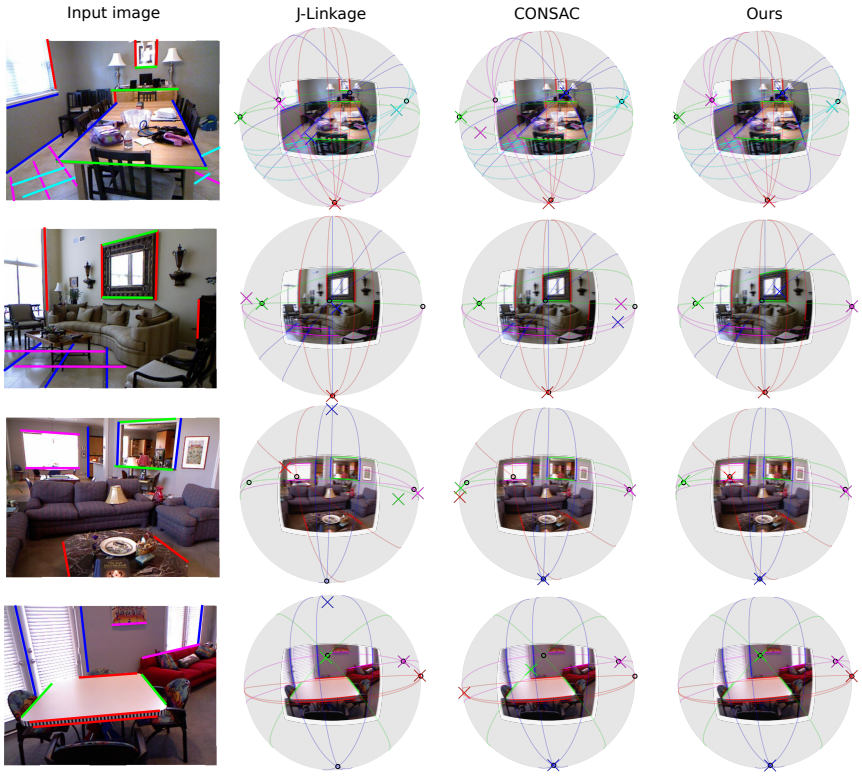


Figure 4.11: **Visualizations on the NYU Depth dataset.** The black ∘ represents the ground truth, while the colored × indicates predictions. Each color corresponds to a set of lines and their related vanishing point. Our model is better at localizing multiple vanishing points in the non-Manhattan world, having predictions (colored cross ×) closer to the ground truth (black ∘), while the predictions of the other methods scatter away from the ground truth, as shown in the first example.

CONSAC sometimes predict vanishing points far away from the ground truth (e.g., the fourth example), because they are originally designed for multiple vanishing point detection in non-Manhattan world, and do not enforce orthogonality explicitly. Ours show better performance in detecting orthogonal vanishing points from complex scenes thanks to the ability to learn semantic features from images directly in an end-to-end manner.

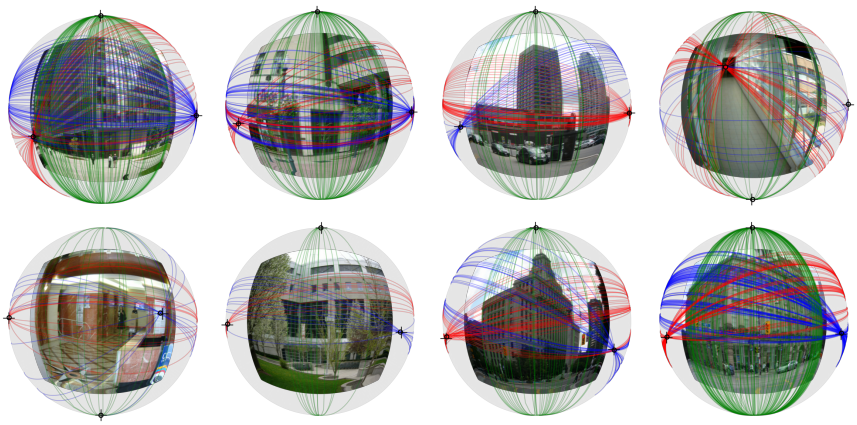


Figure 4.12: **Visualizations on YUD dataset.** We show ground-truth vanishing points (∘) and our predictions (×) on the Gaussian hemisphere, as well as ground truth lines. Our model accurately predicts vanishing points in man-made environments.

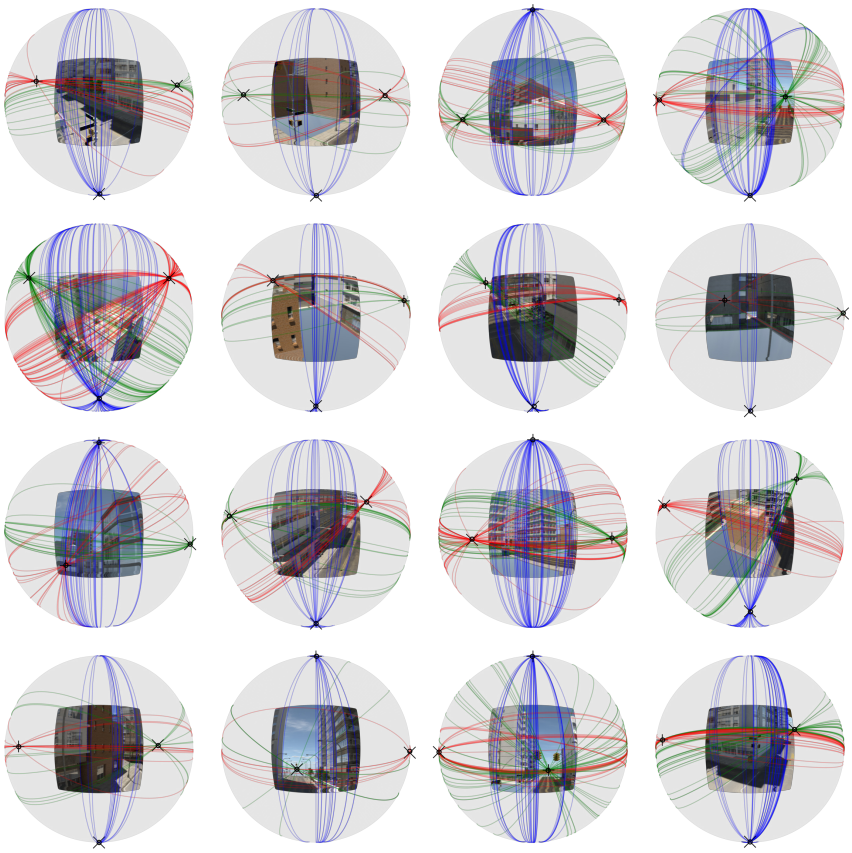


Figure 4.13: **Visualizations on SU3 dataset.** We show ground-truth vanishing points (∘) and our predictions (×) on the Gaussian hemisphere, as well as ground truth lines. Each color represents a cluster of lines that is related to a vanishing point. Our model accurately predicts vanishing points in man-made environments.

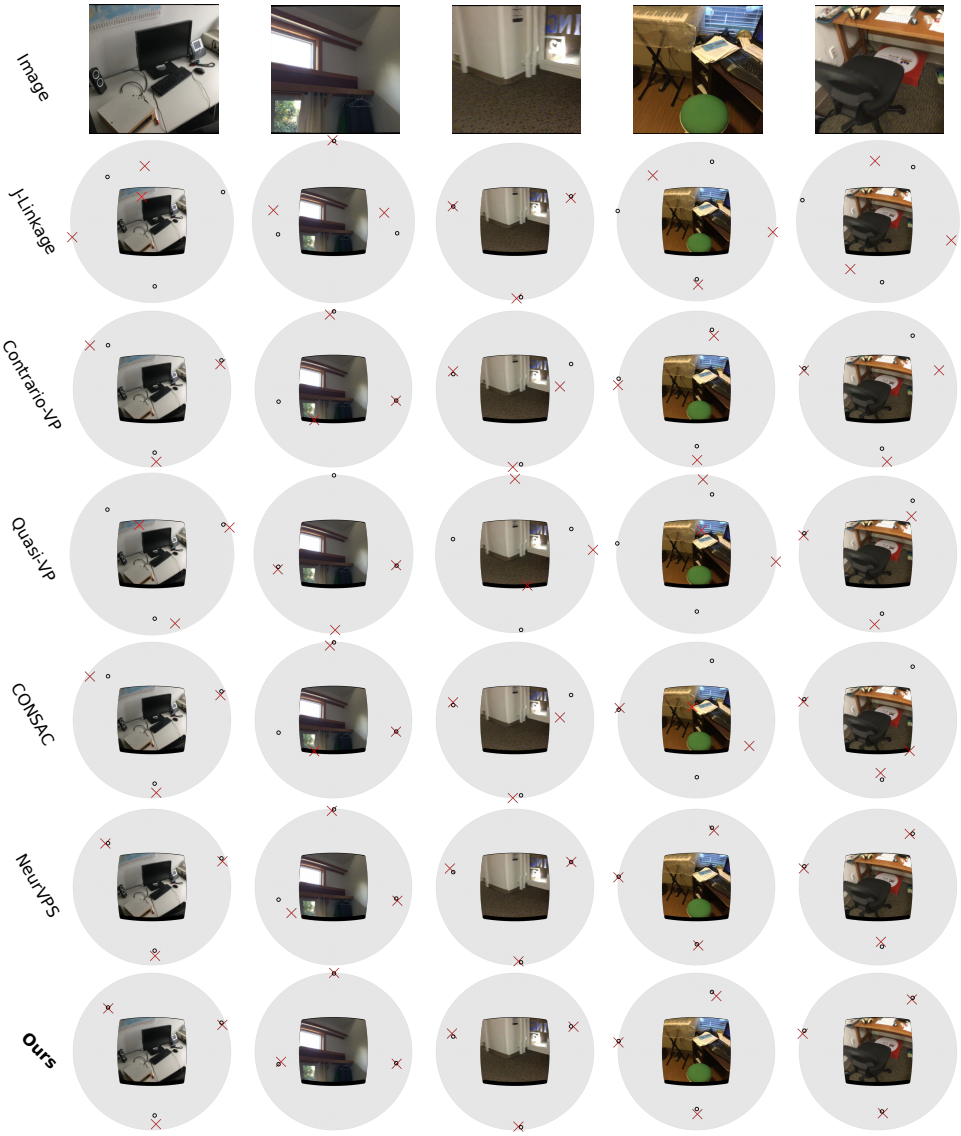


Figure 4.14: **Visualizations on ScanNet dataset.** We show ground-truth vanishing points (○) and predictions from all baseline methods (×) on the Gaussian hemisphere. Learning-based models shows superior performance to classic line segment-based approaches in complex real-world environments.

REFERENCES

[1] R. Cipolla, T. Drummond, and D. P. Robertson, *Camera calibration from vanishing points in image of architectural scenes.* in *BMVC*, Vol. 99 (1999) pp. 382–391.

[2] M. E. Antone and S. Teller, *Automatic recovery of relative camera rotations for urban scenes,* in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, Vol. 2 (IEEE, 2000) pp. 282–289.

[3] L. Grammatikopoulos, G. Karras, and E. Petsa, *An automatic approach for camera calibration from vanishing points,* ISPRS journal of photogrammetry and remote sensing **62**, 64 (2007).

[4] A. Flint, D. Murray, and I. Reid, *Manhattan scene understanding using monocular, stereo, and 3d features,* in *2011 International Conference on Computer Vision* (IEEE, 2011) pp. 2228–2235.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, *Monoslam: Real-time single camera slam,* IEEE transactions on pattern analysis and machine intelligence **29**, 1052 (2007).

[6] H. Li, Y. Xing, J. Zhao, J.-C. Bazin, Z. Liu, and Y.-H. Liu, *Leveraging structural regularity of atlanta world for monocular slam,* in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 2412–2418.

[7] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon, *Vpgnet: Vanishing point guided network for lane and road marking detection and recognition,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 1947–1955.

[8] A. Borji, *Vanishing point detection with convolutional neural networks,* CVPR Scene Understanding Workshop (2016).

[9] C.-K. Chang, J. Zhao, and L. Itti, *Deepvp: Deep learning for vanishing point detection on 1 million street view images,* in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018) pp. 1–8.

[10] X. Zhang, X. Gao, W. Lu, L. He, and Q. Liu, *Dominant vanishing point detection in the wild with application in composition analysis,* Neurocomputing **311**, 260 (2018).

[11] Y. Zhou, H. Qi, J. Huang, and Y. Ma, *Neurvps: Neural vanishing point scanning via conic convolution,* in *Advances in Neural Information Processing Systems* (2019) pp. 866–875.

[12] R. O. Duda and P. E. Hart, *Use of the hough transformation to detect lines and curves in pictures,* Communications of the ACM **15**, 11 (1972).

[13] S. T. Barnard, *Interpreting perspective images,* Artificial intelligence **21**, 435 (1983).

[14] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, *Scannet: Richly-annotated 3d reconstructions of indoor scenes,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 5828–5839.

[15] Y. Zhou, H. Qi, Y. Zhai, Q. Sun, Z. Chen, L.-Y. Wei, and Y. Ma, *Learning to reconstruct 3d manhattan wireframes from a single image,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 7698–7707.

[16] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, *Indoor segmentation and support inference from rgbd images,* in *ECCV* (2012).

[17] Z. Zhou, F. Farhat, and J. Z. Wang, *Detecting dominant vanishing points in natural scenes with application to composition-sensitive image retrieval,* IEEE Transactions on Multimedia **19**, 2651 (2017).

[18] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, *Finding vanishing points via point alignments in image primal and dual domains,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014) pp. 509–515.

[19] E. Lutton, H. Maitre, and J. Lopez-Krahe, *Contribution to the determination of vanishing points using hough transform,* IEEE transactions on pattern analysis and machine intelligence **16**, 430 (1994).

[20] L. Quan and R. Mohr, *Determining perspective structures using hierarchical hough transform,* Pattern Recognition Letters **9**, 279 (1989).

[21] P. L. Palmer and A. Tai, *An optimised vanishing point detector.* in *BMVC* (1993) pp. 1–10.

[22] A. Tai, J. Kittler, M. Petrou, and T. Windeatt, *Vanishing point detection,* in *BMVC92* (Springer, 1992) pp. 109–118.

[23] P. Gamba, A. Mecocci, and U. Salvatore, *Vanishing point detection by a voting scheme,* in *Proceedings of 3rd IEEE International Conference on Image Processing*, Vol. 2 (1996) pp. 301–304.

[24] J. Wu, L. Zhang, Y. Liu, and K. Chen, *Real-time vanishing point detector integrating under-parameterized ransac and hough transform,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021) pp. 3732–3741.

[25] R. Toldo and A. Fusiello, *Robust multiple structures estimation with j-linkage,* in *European conference on computer vision* (Springer, 2008) pp. 537–547.

[26] C. Feng, F. Deng, and V. R. Kamat, *Semi-automatic 3d reconstruction of piecewise planar building models from single image,* CONVR (Sendai:) (2010).

[27] J.-P. Tardif, *Non-iterative approach for fast and accurate vanishing point detection,* in *2009 IEEE 12th International Conference on Computer Vision* (IEEE, 2009) pp. 1250–1257.

[28] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli, *Geometric image parsing in man-made environments,* in *European conference on computer vision* (Springer, 2010) pp. 57–70.

[29] G. McLean and D. Kotturi, *Vanishing point detection by line clustering,* IEEE Transactions on pattern analysis and machine intelligence **17**, 1090 (1995).

[30] F. Schaffalitzky and A. Zisserman, *Planar grouping for automatic detection of vanishing lines and points,* Image and Vision Computing **18**, 647 (2000).

[31] P. Denis, J. H. Elder, and F. J. Estrada, *Efficient edge-based methods for estimating manhattan frames in urban imagery,* in *European conference on computer vision* (Springer, 2008) pp. 197–210.

[32] J. Košecká and W. Zhang, *Video compass,* in *European conference on computer vision* (2002) pp. 476–490.

[33] G. Schindler and F. Dellaert, *Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments,* in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 1 (2004) pp. I–I.

[34] J.-C. Bazin, Y. Seo, C. Demonceaux, P. Vasseur, K. Ikeuchi, I. Kweon, and M. Pollefeys, *Globally optimal line clustering and vanishing point estimation in manhattan world,* in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2012) pp. 638–645.

[35] J.-C. Bazin, Y. Seo, and M. Pollefeys, *Globally optimal consensus set maximization through rotation search,* in *Asian Conference on Computer Vision* (Springer, 2012) pp. 539–551.

[36] K. Joo, T.-H. Oh, J. Kim, and I. S. Kweon, *Robust and globally optimal manhattan frame estimation in near real time,* IEEE transactions on pattern analysis and machine intelligence **41**, 682 (2018).

[37] R. T. Collins and R. S. Weiss, *Vanishing point calculation as a statistical inference on the unit sphere.* in *ICCV*, Vol. 90 (1990) pp. 400–403.

[38] M. J. Magee and J. K. Aggarwal, *Determining vanishing points from perspective images,* Computer Vision, Graphics, and Image Processing **26**, 256 (1984).

[39] M. Straforini, C. Coelho, and M. Campani, *Extraction of vanishing points from images of indoor and outdoor scenes,* Image and Vision Computing **11**, 91 (1993).

[40] M. Zhai, S. Workman, and N. Jacobs, *Detecting vanishing points using global image context in a non-manhattan world,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 5657–5665.

[41] F. Kluger, H. Ackermann, M. Y. Yang, and B. Rosenhahn, *Deep learning for vanishing point detection using an inverse gnomonic projection,* in *German Conference on Pattern Recognition* (Springer, 2017) pp. 17–28.

[42] H. Li, J. Zhao, J.-C. Bazin, and Y.-H. Liu, *Quasi-globally optimal and near/true real-time vanishing point estimation in manhattan world,* IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).

[43] G. Simon, A. Fond, and M.-O. Berger, *A-contrario horizon-first vanishing point detection using second-order grouping laws,* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018) pp. 318–333.

[44] S. Liu, Y. Zhou, and Y. Zhao, *Vapid: A rapid vanishing point detector via learned optimizers,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021) pp. 12859–12868.

[45] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang, *Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 2507–2516.

[46] A. Lengyel, S. Garg, M. Milford, and J. C. van Gemert, *Zero-shot day-night domain adaptation with a physics prior,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021) pp. 4399–4409.

[47] M. Wang and W. Deng, *Deep visual domain adaptation: A survey,* Neurocomputing **312**, 135 (2018).

[48] M. Wulfmeier, A. Bewley, and I. Posner, *Addressing appearance change in outdoor robotics with adversarial domain adaptation,* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017) pp. 1551–1558.

[49] K.-C. Peng, Z. Wu, and J. Ernst, *Zero-shot deep domain adaptation,* in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018) pp. 764–781.

[50] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, *Domain generalization: A survey,* arXiv preprint arXiv:2103.02503 (2021).

[51] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, *Efficient parametrization of multi-domain deep neural networks,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 8119–8127.

[52] Y. Li and N. Vasconcelos, *Efficient multi-domain learning by covariance normalization,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 5424–5433.

[53] M. Antunes and J. P. Barreto, *A global approach for the detection of vanishing points and mutually orthogonal vanishing directions,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013) pp. 1336–1343.

[54] F. M. Mirzaei and S. I. Roumeliotis, *Optimal estimation of vanishing points in a manhattan world,* in *2011 International Conference on Computer Vision* (2011) pp. 2454–2461.

[55] H. Wildenauer and A. Hanbury, *Robust camera self-calibration from monocular images of manhattan worlds,* in *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012) pp. 2831–2838.

[56] F. Kluger, E. Brachmann, H. Ackermann, C. Rother, M. Y. Yang, and B. Rosenhahn, *Consac: Robust multi-model fitting by conditional sample consensus,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 4634–4643.

[57] H. Li, P. Kim, J. Zhao, K. Joo, Z. Cai, Z. Liu, and Y.-H. Liu, *Globally optimal and efficient vanishing point estimation in atlanta world,* in *European Conference on Computer Vision* (Springer, 2020) pp. 153–169.

[58] H. Li, K. Chen, P. Kim, K.-J. Yoon, Z. Liu, K. Joo, and Y.-H. Liu, *Learning icosahedral spherical probability map based on bingham mixture model for vanishing point estimation,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021) pp. 5661–5670.

[59] Á. González, *Measurement of areas on a sphere using fibonacci and latitude–longitude lattices,* Mathematical Geosciences **42**, 49 (2010).

[60] A. Newell, K. Yang, and J. Deng, *Stacked hourglass networks for human pose estimation,* in *European conference on computer vision* (Springer, 2016) pp. 483–499.

[61] Y. Lin, S. L. Pintea, and J. C. van Gemert, *Deep hough-transform line priors, European Conference on Computer Vision*, (2020).

[62] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, *Dynamic graph cnn for learning on point clouds,* Acm Transactions On Graphics (tog) **38**, 1 (2019).

[63] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift,* in *International conference on machine learning* (PMLR, 2015) pp. 448–456.

[64] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, *A density-based algorithm for discovering clusters in large spatial databases with noise.* in *Kdd*, Vol. 96 (1996) pp. 226–231.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, *Scikit-learn: Machine learning in python,* Journal of machine learning research **12**, 2825 (2011).

[66] D. F. Crouse, *On implementing 2d rectangular assignment algorithms,* IEEE Transactions on Aerospace and Electronic Systems **52**, 1679 (2016).

[67] H. Li, J. Zhao, J.-C. Bazin, W. Chen, Z. Liu, and Y.-H. Liu, *Quasi-globally optimal and efficient vanishing point estimation in manhattan world,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 1646–1654.

[68] L. Magri and A. Fusiello, *T-linkage: A continuous relaxation of j-linkage for multi-model fitting,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014) pp. 3954–3961.

[69] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, *Lsd: A fast line segment detector with a false detection control,* IEEE transactions on pattern analysis and machine intelligence **32**, 722 (2008).

[70] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, *Automatic differentiation in pytorch,* (2017).

[71] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

5

Data-Efficient Learning for 3D Mirror Symmetry Detection

We introduce a geometry-inspired deep learning method for detecting 3D mirror planes from single-view images. We reduce the demand for massive training data by explicitly adding 3D mirror geometry in the deep network, as an inductive prior. We extract semantic features, calculate intra-pixel correlations, and build a 3D correlation volume for each plane. The correlation volume indicates the extent to which the input resembles its mirrors at various depth, allowing us to identify the likelihood of each given plane being a mirror plane. Subsequently, we treat the correlation volumes as feature descriptors for planes sampled on a unit hemisphere. Lastly, we design multi-stage spherical convolutions to identify the optimal mirror plane in a coarse-to-fine manner. Experiments on both synthetic and real-world datasets show the benefit of 3D mirror geometry in improving data efficiency and inference speed: up to 25 FPS (on a GeForce RTX 2080 Ti GPU).

This chapter is submitted for publication as:
Yancong Lin, Silvia-Laura Pintea, and Jan van Gemert. *Data-Efficient Learning for 3D Mirror Symmetry Detection*. arXiv: https://arxiv.org/abs/2112.12579.

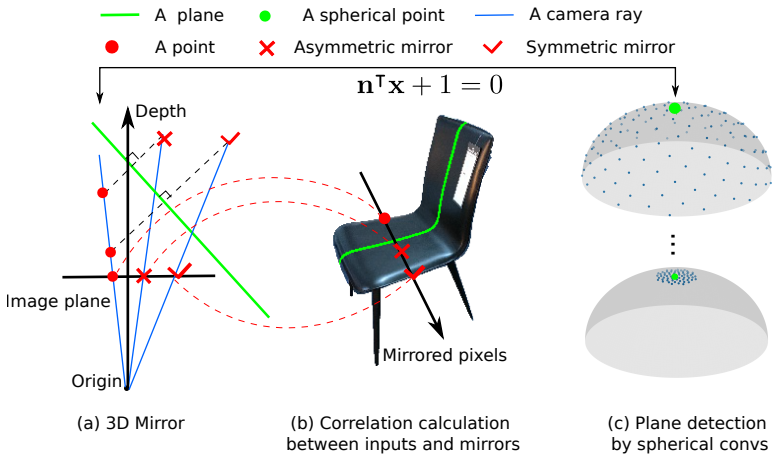


Figure 5.1: **3D mirror symmetry detection.** We identify 3D mirror planes by measuring the correlations between the input and its mirrors over depth. The mirrors are calculated by 3D mirror geometry (a), which localizes the reflections of a given pixel directly on the image plane (b). We describe each candidate plane with feature correlations across points and their mirrors. We sample planes on a hemisphere, and adopt multi-stage spherical convolutions to localize the optimal plane hierarchically (c). The 3D mirror geometry is explicitly encoded in the model. We display the ground truth symmetry axis (in green) on top of the image.

5

5.1. INTRODUCTION

Symmetry exists in nature, in the man-made world, and in science and arts. Mirror symmetry, also known as bilateral or reflection symmetry, is an intrinsic property of many man-made objects, which allows us to infer the entire object from only partial observations, as demonstrated in the shape completion [1, 2] and single-view 3D reconstruction [3–5].

Symmetries can be readily detected in the image plane for frontal-facing objects [6]. However, most images in our daily life are not taken from the front-view and as a result, symmetry in the image plane does not always exist due to the perspective effect. Thus, the inference of 3D mirror symmetries has to handle perspective effects. By relying on local feature matching and camera geometry symmetries can be found on textured objects [7]. More recently, deep learning-based approaches detect mirror symmetry by learning dense features from images [4, 8]. This remedies the effect of unreliable local texture features, and improves the overall performance of symmetry-dependent 3D reconstruction. However, these approaches typically require large annotated datasets, such as ShapeNet [9], for training.

We aim to improve the data efficiency of 3D symmetry detection from a single-view, by explicitly incorporating 3D mirror geometry. Geometrically, a mirror plane, as depicted in green in Figure 5.1, localizes uniquely in the image plane the reflection of every point across this plane, at any given depth. The original points (red circles in Figure 5.1) and their reflections across a candidate symmetry plane (red check-marks in Figure 5.1) should be visually similar for a true symmetry plane. We explicitly incorporate this geometric knowledge into neural networks by characterizing every candidate plane using correlations of features between points and their reflections across depth. We subsequently use these correlations to define a voting space for sampled candidate 3D symmetry planes. We sample these candi-

date 3D symmetry planes in a coarse-to-fine manner on a unit hemisphere and use spherical convolutions to identify the optimal plane from its neighbors. This formulation constraints the 3D symmetry plane prediction to prefer geometrically sound candidates, thus explicitly incorporating geometric priors. These geometric priors no longer need to be implicitly learned from data, which leads to data efficiency. Our work harmonizes the strengths of geometry-based symmetry plane detection with deep learning approaches.

We make the following contributions: (1) We explicitly add 3D mirror geometry into deep networks for finding symmetric correspondences; (2) We design multi-stage spherical convolutions to detect a mirror plane in a coarse-to-fine manner; (3) We improve data efficiency of learning-based 3D mirror plane detection, which has benefits for real-world scenarios where data is scarce; (4) We experimentally demonstrate the added-value of our approach in terms of performance and inference speed on the ShapeNet [9] and Pixel3D [10] datasets.

5.2. RELATED WORK

Planar symmetry detection. A thorough overview on symmetry detection with focus on 2D, is given in [11]. Further work expands on this by including other types of symmetries such as medial-axis-like symmetries and by adding synthetic 3D data [12]. More recently, planar symmetry detection with deep networks for and achieves competitive results [6, 13]. However, for planar symmetry detection objects are typically front-facing, greatly simplifying the task. In addition, planar symmetry does not encode any 3D perspective information. Different from these works, we aim to detect 3D mirror symmetry from single-view images taken from any perspective.

3D mirror symmetry detection. 3D mirror symmetry is prevalent in both nature and the man-made world. There has been excellent research on utilizing geometric transforms for detecting mirror symmetries from 3D inputs [14–16]. A 3D Hough transform proves effective at detecting mirror symmetry planes from point clouds, in [14]. Alternatively, planar reflective symmetry transform can find symmetry planes in 3D volumetric data [16]. Similar to these works, we also make use of the 3D geometric knowledge for detecting mirror symmetries, but instead or relying on 3D data we start from single-view images.

Recently, deep networks have been used for leveraging large datasets for learning 3D symmetries [1, 17]. Despite being able to detect multiple symmetries, they rely on heavy post-processing procedures to find the optimal symmetry. Moreover, these models have only been tested on synthetic 3D datasets (i.e. ShapeNet [9] with voxelized volumes or RGB-D data). In contrast, we propose to learn 3D mirror symmetry in an end-to-end manner, and test on both synthetic and real-world 2D images.

3D mirror symmetry from single-view images. A 2-stage approach can be effective for 3D mirror symmetry detection from 2D images, by first matching image correspondences and then applying RANSAC to identify the best symmetry plane [7]. However, this strategy is no longer applicable in the absence of texture, or on smooth surfaces, or repetitive patterns, because of incorrect correspondences. Rather than relying on local feature matching, NeRD [8] makes use of neural networks to learn dense features and incorporates 3D mirror geometry into learning, making it the top-performing model. Similarly, we also explicitly add geometric knowledge into learning. However, different from NeRD which relies on

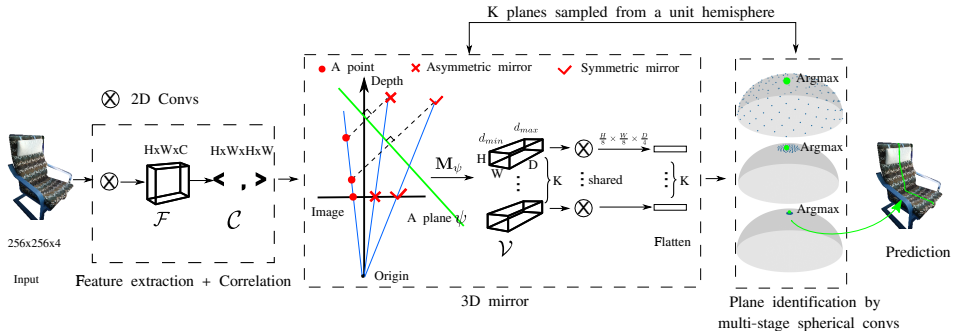


Figure 5.2: **Overview.** Our model includes three components: feature extraction, correlation calculation, and 3D mirror and plane identification by spherical convolutions. We first calculate intra-pixel correlations $\mathcal{C}$ using learned features $\mathcal{F}$. Then we build a correlation volume $\mathcal{V}$ for each sampled plane ψ, which we then flatten as a feature descriptor. We adopt spherical convolutions on uniformly sampled planes on a hemisphere to locate the optimal plane (highlighted in green).

5

large amounts of training data, we reduce the inference latency and the demand for massive annotated data, by calculating the correlations between the inputs and its mirrors and by introducing multi-stage spherical convolutions to the localize the optimal mirror plane.

5.3. GEOMETRIC PRIORS FOR 3D MIRROR SYMMETRY

Our method starts from RGBA images as input and outputs sampled planes and associated confidence sores for being a mirror plane. We choose the plane with the highest confidence as our prediction. Although an object may admit multiple symmetries, we only predict the principal mirror symmetry, as in [8]. We explicitly add 3D mirror geometry knowledge into deep networks to improve the data efficiency. Our model is composed of three parts: (1) feature extraction and correlation calculation, (2) 3D mirroring, (3) plane identification by spherical convolutions, as in Figure 5.2.

5.3.1. FEATURE EXTRACTION AND CORRELATION CALCULATION

We start by learning semantic image features via a convolutional neural network as in [8], which results in a feature map $\mathcal{F}$ of size $[H \times W \times C]$, where H, W, C indicate height, width, and number of channels, respectively. Then, we calculate the intra-pixel correlation between each pair of points in the $[H \times W]$ grid by dot product over the channel dimension. This produces a correlation tensor $\mathcal{C}$ of size $[H \times W \times H \times W]$. $\mathcal{C}$ indicates the extent to which a pixel resembles the others. A higher correlation implies higher visual similarity. In the convolutional network we subsample the featuremaps to size [64× 64] and set C to 64.

5.3.2. 3D SYMMETRIES

We input the correlation tensor $\mathcal{C}$ into the 3D mirror module, together with a sampled plane. This module outputs a feature descriptor for each considered mirror plane. Given a randomly sampled plane ψ, the 3D mirror module computes the symmetric correspondences $\mathbf{x}''$ for each pixel $\mathbf{x}'$ across various depths d. The module then aggregates the correlations per pixel over all its correspondences in a 3D correlation volume of size $[H \times W \times D]$. At

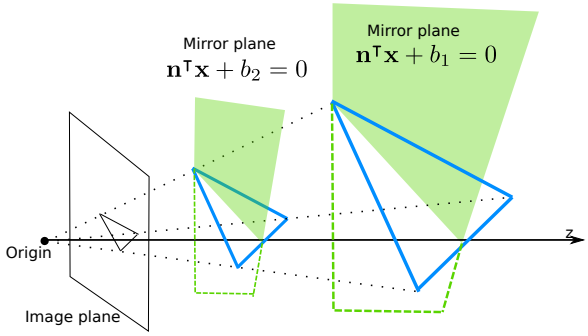


Figure 5.3: **Scale ambiguity.** The two objects (in blue) only differ in scale, but their projections on the image plane are the same. Therefore, we are unable to determine the scale of the object, or the actual value of the offset. In practice, we set $b = 1$.

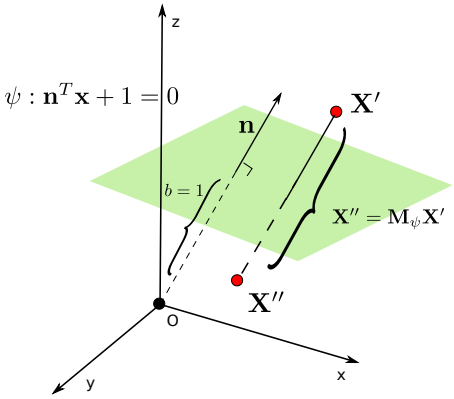


Figure 5.4: **3D mirror geometry.** 3D points $\mathbf{X}'$ and $\mathbf{X}''$ are symmetric with respect to the given plane ψ defined by $\mathbf{n}^\intercal\mathbf{x} + 1 = 0$. We represent $\mathbf{X}' \in \mathbb{R}^4$ and $\mathbf{X}'' \in \mathbb{R}^4$ in the homogeneous coordinate, where $\mathbf{X}'' = \mathbf{M}_\psi\mathbf{X}'$ as defined in Eq. (5.1). $\mathbf{M}_\psi \in \mathbb{R}^{4\times4}$ is the 3D mirror transformation uniquely determined by $\mathbf{n}$, allowing us to calculate mirrors explicitly [7, 14].

every point, the correlation volume indicates how similar the mirrors across the sampled plane ψ are to that point. A higher similarity indicates that the sampled plane ψ is more likely to be the optimal mirror plane. Subsequently, we downsample the correlation volume and flatten it into a 1D feature vector that describes the currently sampled plane, ψ.

(i) 3D mirror planes. A plane ψ is uniquely defined by its normal direction $\mathbf{n} \in \mathbb{R}^3$ and offset $b \in \mathbb{R}$ as $\mathbf{n}^\intercal\mathbf{x} + b = 0$, where $\mathbf{x} \in \mathbb{R}^3$ denotes points on the plane. We are unable to determine b due to scale ambiguity [8, 18]. This is because the scene can be moved arbitrarily along the normal direction $\mathbf{n}$ and scaled accordingly, without affecting the image, as shown in Figure 5.3. Therefore, we only need to predict the normal direction $\mathbf{n}$ of the mirror plane. Moreover, given that a normal direction $\mathbf{n}$ is equivalent to a point on a unit hemisphere, we can further define a plane as a spherical point. Thus we can sample planes on a unit hemisphere.

(ii) 3D mirror transform. Figure 5.4 shows an illustration of 3D mirror geometry for a randomly sampled plane ψ defined by $\mathbf{n}^\top\mathbf{x}+1=0$. The corresponding 3D mirror transformation $\mathbf{M}_\psi \in \mathbb{R}^{4\times4}$ associated to plane $\psi : \mathbf{n}^\top\mathbf{x}+1=0$ is uniquely defined by the normal direction of the plane $\mathbf{n}$, as in Eq. (5.1) coordinate[7, 14]:

$$\mathbf{X}'' = \underbrace{\begin{pmatrix}\mathbf{I}-2\mathbf{n}\mathbf{n}^\top & -2\mathbf{n}\\ \mathbf{0} & 1\end{pmatrix}}_{\mathbf{M}_\psi}\mathbf{X}', \tag{5.1}$$

where $\mathbf{X}'' \in \mathbb{S}$ and $\mathbf{X}' \in \mathbb{S}$ are a pair of symmetric 3D points, and $\mathbb{S} \subset \mathbb{R}^4$ is the set of 3D points on the object surface in homogeneous coordinates.

Given the camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{4\times4}$, we can project both $\mathbf{X}'$ and $\mathbf{X}''$ on the image plane by $\mathbf{x}' = \mathbf{K}\mathbf{X}'/d'$ and $\mathbf{x}'' = \mathbf{K}\mathbf{X}''/d''$, where d' and d'' are the corresponding depths in the camera space. Therefore, we can derive the constraint between points $\mathbf{x}'$ and their projections $\mathbf{x}''$ as:

$$\mathbf{x}''d'' = \mathbf{K}\mathbf{M}_\mathbf{P}\mathbf{K}^{-1}\mathbf{x}''d'', \tag{5.2}$$

where $\mathbf{x}' = [x', y', 1, 1/d']$ and $\mathbf{x}'' = [x'', y'', 1, 1/d'']$ indicate the coordinates of the projected points in the pixel space. Eq. (5.2) enables us to find the symmetric correspondences of every pixel at various depths, given a sampled mirror plane, ψ.

(iii) 3D correlation volume. We follow Eq. (5.2) to localize the symmetric correspondences (x'', y'') for each pixel (x', y') at various depth $d \in \mathcal{D}$, where $\mathcal{D} = \{d_{min} + \frac{i}{D-1}(d_{max} - d_{min}) | i = 0, 1, ..., D-1\}$. d_{min} and d_{max} are the minimal and maximal depth values. Subsequently, we index the correlation tensor at $\mathcal{C}(x', y', x'', y'')$ via bi-linear interpolation. Lastly, we fill in $\mathcal{V}(x', y', d)$ with the indexed value. We enumerate all $d \in \mathcal{D}$ for each pixel (x', y') in the $[H \times W]$ grid, thus obtaining a correlation volume $\mathcal{V}$ of size $[H \times W \times D]$.

The correlation volume $\mathcal{V}$ indicates the similarity between each input and its mirrors at all sampled depths. This enables deep networks to learn if there exists a visually similar reflection for a given plane. A higher similarity implies that the give plane is more likely to be a mirror plane. However, to identify the global mirror plane, we need to aggregate information over the entire $\mathcal{V}$. Otherwise, the model may only predict local symmetries. To this end, we apply 2D convolutions to downscale $\mathcal{V}$, resulting in an output tensor of size $[\frac{H}{8} \times \frac{W}{8} \times \frac{D}{4}]$. We, then, flatten the downscaled output to a 1D vector for further classification. One such 1D vector describes each plane sampled on the hemisphere.

5.3.3. PLANE DETECTION BY SPHERICAL CONVOLUTIONS

To pinpoint the optimal mirror plane, we need an exhaustive search on the entire hemisphere, which is computationally infeasible. Instead, we adopt a multi-stage sampling strategy using Fibonacci lattice [19] in a coarse-to-fine manner. In practice, we sample planes over 3 stages to reach a desirable precision. $\mathbb{P}_i = \{\mathbf{n}_i^k\}_{k=1}^K \subset \mathbb{R}^3$ represents all sampled planes at ith stage, where $\mathbf{n}$ is the normal direction, and K is the number of planes. This results in a spherical point cloud of size K at each stage, where each spherical point (plane) has an associated with an 1D feature descriptor. Since all K planes at each stage are uniformly sampled, we can utilize spherical convolutions to identify the most probable candidate by comparing a plane with its neighbors. We use EdgeConv [20] to convolve at each stage. The EdgeConv module extracts features from a local neighborhood with a

fully connected layer, a BatchNorm layer [21] and a LeakyReLU activation. We treat each sampled point as a node, and compute its top 16 nearest neighbors.

The multi-stage sampling differs in training and inference. During training, we uniformly sample K planes $\mathbb{P}_i = \{\mathbf{n}^k : \arccos(|\langle \mathbf{n}^k, \mathbf{n}^* \rangle|) \leq \delta_i\}_{k=1}^K$ on the hemisphere at the i^{th} stage and assign the nearest neighbor to the ground truth $\mathbf{n}^*$ as the label. δ_i is the scale factor which defines the size of sampling region at the i^{th} stage. We minimize the binary cross-entropy loss at each stage, averaged over the positive and negative samples respectively due to the class imbalance. During inference, we start with a set of uniformly sampled K planes $\mathbb{P}_1 = \{\mathbf{n}^k\}_{k=1}^K \subset \mathbb{R}^3$ on the whole hemisphere. Subsequently, we take from the previous step the best prediction $\hat{\mathbf{n}}_{i-1}$ as the center and sample another K planes $\mathbb{P}_i = \{\mathbf{n}^k : \arccos(|\langle \mathbf{n}^k, \hat{\mathbf{n}}_{i-1} \rangle|) \leq \delta_i\}_{k=1}^K$. Finally, we choose the plane with the highest estimated confidence as our prediction.

5.4. EXPERIMENTS

5.4.1. EXPERIMENTAL SETUP

Datasets. We conduct experiments on ShapeNet [9] and Pixel3D [10]. The objects in both datasets are aligned to the canonical space such that the Y-Z plane is the 3D mirror symmetry plane. On the synthetic ShapeNet dataset, we use the same subset as in [8] for fair comparison. There are 175,122/500/8,756 images in the training/validation/test splits, respectively. All images are of size 256×256 pixels. On the real-world Pixel3d dataset, we also follow [8] to pre-process the data. We first crop the images to obtain the objects inside bounding boxes. Sequentially, we rescale them to 256×256, and adjust the camera intrinsic matrix $\mathbf{K}$ accordingly. This results in a dataset of 5,285 and 588 images for training and test respectively.

Evaluation. We follow [8] and evaluate all methods by measuring the angle difference of the plane normals between the ground-truth and predictions in the camera space. We calculate the percentage of the predictions that have a smaller angle difference than a given threshold and plot the angle accuracy (AA) curves.

Implementation details. We implement our model in Pytorch [22], and provide our code online. The $x = 0$ plane in the object space is considered as the ground truth because it is explicitly aligned for each object [9]. We set $d_{min} = 0.64$, $d_{max} = 1.23$ and $D = 64$ for depth. We perform spherical convolutions at 3 scales and sample $128, 64, 64$ symmetry planes at each scale. We set the scale factor to be $\delta = \{90.0°, 12.86°, 3.28°\}$. All models are trained from scratch on each dataset on Nvidia RTX2080 GPUs with the Adam optimizer [23], for 32 epochs at most. The learning rate and weight decay are set to be 3×10^{-4} and 1×10^{-7}. We decay the learning rate by 10 after 24 epochs. To maximize the GPU usage, we set batch size to 6. The inference speed is approximately 25 FPS.

Baselines. We compare our model primarily with NeRD [8], the state-of-the-art work on 3D mirror symmetry detector, in all experiments. We also implement a simple baseline using direct regression to estimate the symmetry normal $\mathbf{n}$. We implement this baseline on top of a ResNet-50 [24] backbone with L_1 loss. Front2Back [3] also detects 3D mirror symmetry using a variant of classical iterative closest point approach. However, it requires depth maps in advance and has only been tested on ShapeNet. We also compare with RotCon

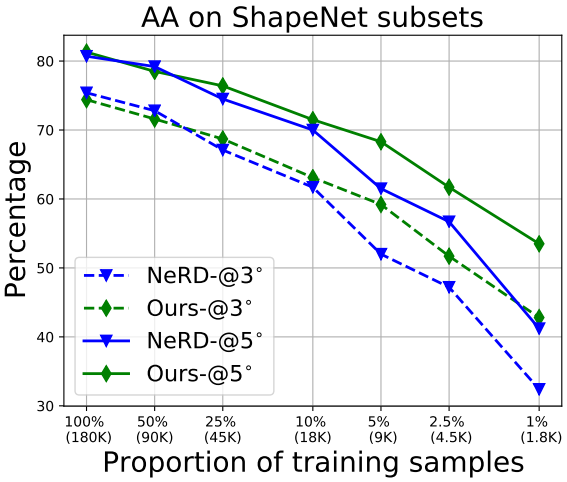


Figure 5.5: **Exp1: Data efficiency**. Comparison between our model and NeRD when training on various ShapeNet subsets. Both models are trained from scratch on each subset. The difference of the two models is limited when the training data is ample (e.g., $18K \geq$). However, our model outperforms NeRD when training on limited data (e.g. $\leq 9K$ images). This gain is especially visible on the 1% subset where the AA difference is more than 10%.

[25] which proposes a continuous representation for estimating 3D rotation. We use L_1 loss for training and report its performance on both datasets. DISN [4] learns 6D rotation representation for estimating camera poses on ShapeNet. We recover the normal of the mirror plane from camera poses and report the performance of their pre-trained models on ShapeNet. NCOS [26] defines a *normalized object coordinate space (NOCS)* and identifies 6D representations of camera poses. We use NOCS to estimate the orientation of objects on ShapeNet.

5.4.2. EXP 1: DATA EFFICIENCY

We evaluate the data efficiency of our model by reducing the number of training samples to $\{50\%, 25\%, 10\%, 5\%, 2.5\%, 1\%\}$ on the ShapeNet dataset, which has approximately 200K training images in total. We train all models from scratch and compare the AA scores at 3° and 5° on the complete test set. We compare our model with NeRD which holds state-of-the-art result, as shown in Figure 5.5. The two models have similar amount of parameters, thus removing the impact of parameters. In general, our model shows superiority over NeRD with the decrease of training samples, and this advantage accentuates on subsets with fewer than 10K images. When training on $100\% - 10\%$ subsets, we observe marginal difference, while on $10\% - 1\%$ subsets, we observe a drastic difference (up to 10% in AA), indicating our model is more efficient in learning from limited data. Notably, our model can achieve similar results as NeRD with only half of the training data as seen on the 2.5% and 5% subsets, thus demonstrating the data efficiency of our model.

5.4.3. EXP 2: COMPARISON WITH STATE-OF-THE ART

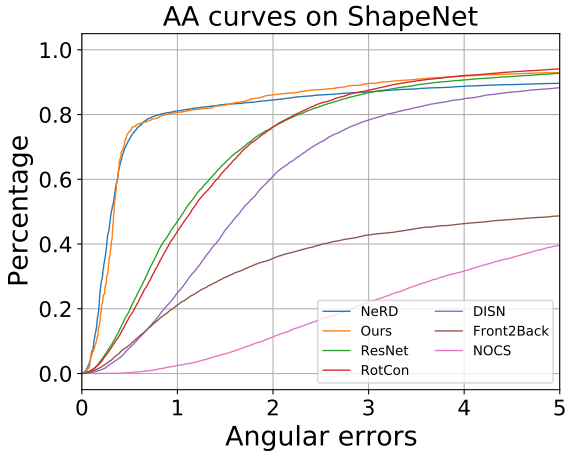


Figure 5.6: **Exp 2.1: Comparison on synthetic ShapeNet.** Thanks to the bake-in 3D mirror geometry, NeRD and our model outperform other models substantially, particularly in low-error region.

Datasets	ShapeNet[9]		
Metrics	AA@1°	AA@3°	AA@5°
ResNet [24]	20.8	55.7	69.5
RotCon [25]	18.7	54.6	69.4
DISN [4]	9.3	26.1	34.1
NOCS [26]	0.6	7.9	17.3
Front2Back [3]	9.3	26.1	34.1
NeRD [8]	**57.3**	75.4	80.7
Ours	55.7	**75.5**	**82.0**

Table 5.1: **Exp 2.1: Evaluation on synthetic ShapeNet.** Our model performs competitively on the synthetic ShapeNet dataset. Moreover, our model is approximately ×20 faster than the top-performing NeRD during inference (25FPS *vs* 1.4FPS).

EXP 2.1: COMPARISON ON SYNTHETIC DATA

Figure 5.6 and Table 5.1 shows the comparison with state-of-the-art models on the synthetic ShapeNet dataset. Our model displays competitive results and enjoys a considerable advantage in the low-error region. The prediction error of our model is less than 1° in 80% of the test cases. NeRD performs the best but is approximately × 20 slower than our model during inference (1.4 *vs* 25 FPS). In comparison, the ResNet baseline using direct regression can only reach approximately 50% AA at 1°, indicating that naive convolutions lack the ability to exploit the mirror symmetry, even with ample training data. We also notice that end-to-end approaches outperform models relying on heavy post-processing, such as Front2Back [3].

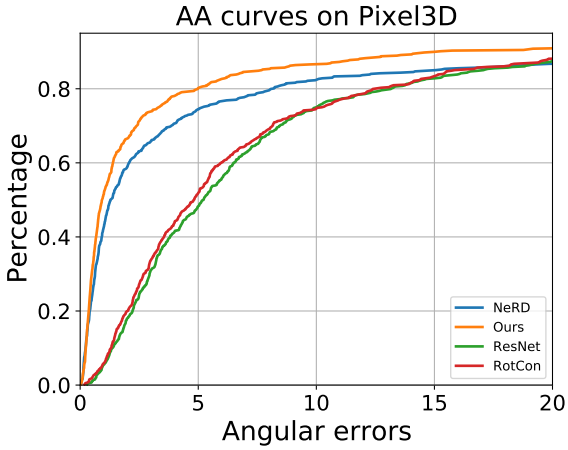


Figure 5.7: **Exp 2.2: Comparison on real-world Pixel3D dataset.** Our model performs the best on the challenging real-world dataset, as the knowledge of 3D mirror no longer needs to be learned from massive data.

Datasets	Pixel3D[10]		
Metrics	AA@1°	AA@5°	AA@10°
ResNet [24]	1.5	12.3	23.6
Rotcon [25]	2.2	14.1	25.8
NeRD [8]	22.7	46.0	55.8
Ours	**27.5**	**53.0**	**62.8**

Table 5.2: **Exp 2.2: Evaluation on real Pixel3D dataset.** Our model performs the best on the real-world Pixel3D dataset.

EXP 2.2: COMPARISON ON REAL-WORLD DATA

To further validate the effectiveness of our model, we also test on the real-world Pixel3D dataset [10], as shown in Figure 5.7 and Table 5.2. Our model outperforms all the other models consistently, thus demonstrates the superiority of our design. It is worth noting that the prediction error on Pixel3D is relatively larger than on ShapeNet. On one hand, there is limited training data (5,000 images in total), which is significantly less than ShapeNet. On the other hand, the camera configuration differs from image to image, thus making it hard to make precise predictions. Our model incorporates camera intrinsics into the 3D mirror geometry, and therefore makes better predictions in different camera settings. Meanwhile, the usage of spherical convolutions contributes to the outstanding results on Pixel3D. NeRD lags behind due to a high demand for training data.

5.4.4. EXP 3: ABLATION STUDIES

To verify the contribution of each component in our design, we conduct ablation studies, as shown in Table 5.3. All models are trained on the ShapeNet 1% subset. Model (a) is a simple baseline using direct regression and shows inferior results to the others in detecting

	3D Mirror	Correlation volumes	Spherical convs	AA@1°	AA@5°
a	✗	✗	✗	0.8	9.5
b	✓	✓	✗	15.8	44.4
c	✓	✗	✓	8.0	31.5
d	✓	✓	✓	**22.1**	**53.5**

Table 5.3: **Exp 3: Ablation studies.** We quantitatively verify the added value of 3D mirror geometry, 3D correlation volumes, and spherical convolutions. All these design choices are essential for the performance of our model.

3D mirror geometry. We replace the spherical convolutions in our design (d) with 1×1 convolutions in model (b). Comparing (b) and (d), we find that spherical convolutions improve the results significantly. In model (c), we replace the convolutions over the 3D cost volumes $\mathcal{V}$ by taking the max over the depth dimension. By doing so, we only obtain the correspondence with the highest correlation across different depths for each pixel, thus removing the 3D spatial information. However, model (c) substantially underperforms model (d), thus validating the necessity of 3D cost volumes. The ablation studies justify the added value of the 3D mirror, cost volumes, and spherical convolutions.

5.4.5. DISCUSSION AND LIMITATION

We compare the predictions from NeRD and our model qualitatively in this section. We rely on ground truth depth for visualizing the mirror symmetry on the image plane. Figure 5.8 compares NeRD (middle) and our model (right), when training on the 1% subset only. The green line shows the ground truth symmetry axis on the image plane. We highlight in red the errors between the prediction and the ground truth. The comparison verifies the advantage of our model in small data regime where only 1% data is available.

Figure 5.9 shows the impact of adding more training data from 1%, 10% to 100%. In the leftmost examples, our model fails on the 1% subset, as it predicts a plane orthogonal to the ground truth. However, our model can make effective use of additional training data and recover from substantial mistakes. The examples on the right show that also for our model adding data improves precision qualitatively. We guide the reader to the supplementary material for more visualizations.

We show the failure cases of our model in Figure 5.10. The main challenge in these examples is that an object may admit multiple symmetries, thus leading to ambiguity. Moreover, it can be hard to find symmetric correspondences for certain objects, such as the gun example. One of the drawbacks of our models is that if the found point correspondences are not sufficiently similar in appearance, the mirror plane detection will be erroneous. Additionally, for now our model is limited to only detecting one primary symmetry plane, and this is restrictive as certain objects may display multiple symmetries.

5.5. CONCLUSION

This paper studies 3D mirror symmetry detection from single-view perspective images, replying on well-founded geometric priors. We explicitly incorporate 3D mirror geometry

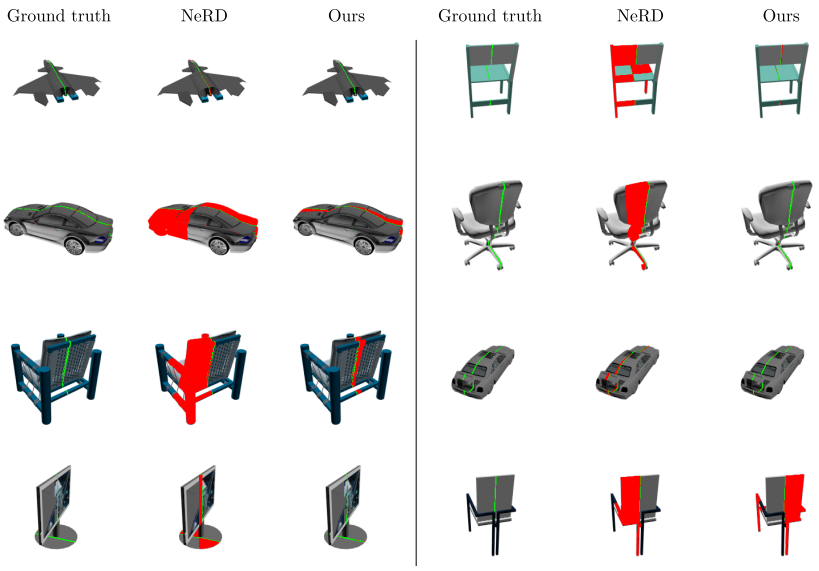


Figure 5.8: **Qualitative results on ShapeNet when training on the** 1% **subset**. We compare the ground truth, predictions from NeRD and our results when training on the 1% subset only. We highlight the ground truth symmetry axis in green and prediction errors in red. In general, our model shows superior results over NeRD in small data regime. However, both models fail on the example in the bottom-left corner due to the lack of correspondences.

into learning and propose multi-stage spherical convolutions to locate the optimal mirror plane precisely and efficiently. Our model is fully end-to-end trainable. Extensive experiments on both synthetic and real-world datasets shows the superiority of our model in both improved data efficiency and state-of-the-art performance. However, our model can only predict the primary mirror symmetry plane. Future work will focus on extending our model for detecting multiple types of symmetries, such as rotation symmetry and translation symmetry. Exploring the usage of 3D mirror in single-view 3D reconstruction, such as depth estimation and shape completion, is also a promising future research direction.

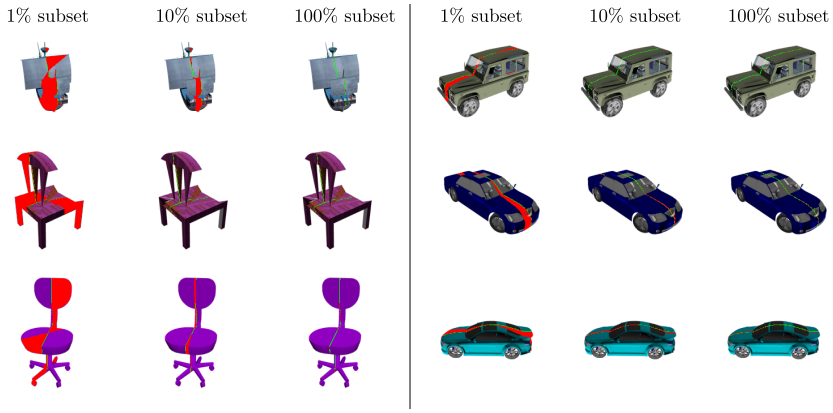


Figure 5.9: **Qualitative results on ShapeNet when adding more training data**. We compare the predictions from our models trained on the 1%, 10% and 100% (sub)sets. The green line represents the ground truth symmetry axis while the red region indicates errors. Our model can effectively take advantage of additional data and thus recovers from making substantially wrong predictions. (See supplementary material for more examples)

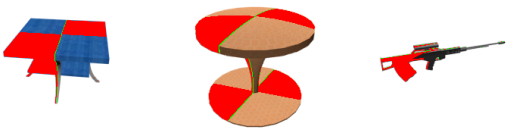


Figure 5.10: **Failure cases on ShapeNet.** Our model fails in certain scenarios due to the presence of multiple symmetries and lack of reliable correspondences. We highlight the ground truth in green and prediction errors in red.

5.6. SUPPLEMENTARY MATERIAL

We also display results from the real-world Pixel3D dataset [10] in Figure 5.11. In general, our model is able to detect the dominant mirror symmetry accurately from images unless multiple symmetries are present.

Figure 5.11: **Qualitative results on Pixel3D.** Our model is able to detect the dominant mirror symmetry from real-world images in most cases. However it fails to handle multiple symmetries (shown in the rightmost column). The ground truth symmetry axis is in green, while the prediction errors are in red.

REFERENCES

[1] L. Gao, L.-X. Zhang, H.-Y. Meng, Y.-H. Ren, Y.-K. Lai, and L. Kobbelt, *Prs-net: Planar reflective symmetry detection net for 3d models,* IEEE Transactions on Visualization and Computer Graphics **27**, 3007 (2020).

[2] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, *Morphing and sampling network for dense point cloud completion,* in *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34 (2020) pp. 11596–11603.

[3] Y. Yao, N. Schertler, E. Rosales, H. Rhodin, L. Sigal, and A. Sheffer, *Front2back: Single view 3d shape reconstruction via front to back prediction,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 531–540.

[4] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, *Disn: Deep implicit surface network for high-quality single-view 3d reconstruction,* arXiv preprint arXiv:1905.10711 (2019).

[5] S. Wu, C. Rupprecht, and A. Vedaldi, *Unsupervised learning of probably symmetric deformable 3d objects from images in the wild,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 1–10.

[6] A. Seo, W. Shim, and M. Cho, *Learning to discover reflection symmetry via polar matching convolution,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021) pp. 1285–1294.

[7] K. Köser, C. Zach, and M. Pollefeys, *Dense 3d reconstruction of symmetric scenes from a single image,* in *Joint Pattern Recognition Symposium* (Springer, 2011) pp. 266–275.

[8] Y. Zhou, S. Liu, and Y. Ma, *Nerd: Neural 3d reflection symmetry detector,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 15940–15949.

[9] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, *Shapenet: An information-rich 3d model repository,* arXiv preprint arXiv:1512.03012 (2015).

[10] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman, *Pix3d: Dataset and methods for single-image 3d shape modeling,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) pp. 2974–2983.

[11] J. Liu, G. Slota, G. Zheng, Z. Wu, M. Park, S. Lee, I. Rauschert, and Y. Liu, *Symmetry detection from realworld images competition 2013: Summary and results,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2013) pp. 200–205.

[12] C. Funk, S. Lee, M. R. Oswald, S. Tsogkas, W. Shen, A. Cohen, S. Dickinson, and Y. Liu, *2017 iccv challenge: Detecting symmetry in the wild,* in *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2017) pp. 1692–1701.

[13] C. Funk and Y. Liu, *Beyond planar symmetry: Modeling human perception of reflection and rotation symmetries in the wild,* in *Proceedings of the IEEE International Conference on Computer Vision* (2017) pp. 793–803.

[14] D. Cailliere, F. Denis, D. Pele, and A. Baskurt, *3d mirror symmetry detection using hough transform,* in *2008 15th IEEE International Conference on Image Processing* (IEEE, 2008) pp. 1772–1775.

[15] K. Sfikas, T. Theoharis, and I. Pratikakis, *Rosy+: 3d object pose normalization based on pca and reflective object symmetry with application in 3d object retrieval,* International Journal of Computer Vision **91**, 262 (2011).

[16] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, *A planar-reflective symmetry transform for 3d shapes,* in *ACM SIGGRAPH 2006 Papers* (2006) pp. 549–559.

[17] Y. Shi, J. Huang, H. Zhang, X. Xu, S. Rusinkiewicz, and K. Xu, *Symmetrynet: Learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images,* ACM Transactions on Graphics (SIGGRAPH Asia 2020) **39** (2020).

[18] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*, Vol. 26 (Springer Science & Business Media, 2012).

[19] Á. González, *Measurement of areas on a sphere using fibonacci and latitude–longitude lattices,* Mathematical Geosciences **42**, 49 (2010).

[20] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, *Dynamic graph cnn for learning on point clouds,* Acm Transactions On Graphics (tog) **38**, 1 (2019).

[21] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift,* in *International conference on machine learning* (PMLR, 2015) pp. 448–456.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, and A. Antiga, *L. and lerer,* Automatic differentiation in pytorch (2017).

[23] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization,* arXiv preprint arXiv:1412.6980 (2014).

[24] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[25] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, *On the continuity of rotation representations in neural networks,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 5745–5753.

[26] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, *Normalized object coordinate space for category-level 6d object pose and size estimation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 2642–2651.

6

Investigating Transformers in the Decomposition of Polygonal Shapes as Point Collections

Transformers can generate predictions in two approaches: 1. auto-regressively by conditioning each sequence element on the previous ones, or 2. directly produce an output sequences in parallel. While research has mostly explored upon this difference on sequential tasks in natural language processing, we study the difference between auto-regressive and parallel predictions on visual set prediction tasks, and in particular on polygonal shapes in images because polygons are representative of numerous types of objects, such as buildings or obstacles for aerial vehicles. This is challenging for deep learning architectures as a polygon can consist of a varying carnality of points. We provide evidence on the importance of natural orders for Transformers, and show the benefit of decomposing complex polygons into collections of points in an auto-regressive manner.

This chapter is published as:
Andrea Alfieri, **Yancong Lin** and Jan van Gemert. Investigating Transformers in the Decomposition of Polygonal Shapes as Point Collections. International Conference on Computer Vision *workshops* (ICCVW), 2021. arXiv: https://arxiv.org/abs/2108.07533.

6.1. INTRODUCTION

Predicting polygonal shapes in images is a high-level task that is representative of many vision problems. One example is to automatically extract vectorized building outlines from high-resolutions satellite images [1–4]. Similarly, polygon detection can be beneficial for vision-based flight control of unmanned aerial vehicles (UAVs), as demonstrated in the Autonomous Drone Racing [5], where drones fly autonomously through a sequence of polygon-shaped gates purely relying on vision signals. The performance of polygon detection is critical for UAVs as it directly affects the precision of navigation.

In this work, we treat polygonal shape prediction as a *collection prediction* task such as sequences and sets . Many kinds of data can be naturally represented using collection and many machine learning tasks can be viewed as a collection prediction problem, such as predicting the collection of points of a polygon [6], detecting objects in an image [7, 8], estimating the pose of humans by detecting a set of key-points [9, 10], or predicting multiple labels for the same sample [11]. The difficulty in such problems arises for two main reasons: because the *cardinality* (i.e. the number of elements) of the collection is unknown and can vary among different samples, and because collections when treated as a set are *permutation-invariant*. However, canonical convolutional neural networks are not able to tackle these problems by design [12].

Transformers [13] have achieved impressive results on numerous collection prediction tasks [7, 12], thanks to the attention module [14], a solution capable of aggregating information from the entire collection while also satisfying permutation invariance.

The work in [13] first presents Transformers as an auto-regressive sequence-to-sequence model that generates a collection of output tokens one by one. Numerous following works [15–17] introduce new variants of Transformers to reduce compute latency. However, the vast majority of auto-regressive Transformers focuses on sequential tasks only, such as machine translation [16, 17] or speech recognition [15]. Recently, parallel Transformers have successfully been applied into computer vision, especially in object detection [7, 18], where the model outputs a set of bounding boxes *in parallel*. However, parallel Transformers rely on computationally intensive strategies, such as oversampling and Hungarian matching [19].

The difference between auto-regressive and parallel Transformers is fundamental. Let us consider an image containing a collection of three objects (namely A, B and C) which we are trying to detect. When asking a parallel Transformer to learn this task, we are actually asking the model to learn the joint probability of these three variables, conditioned on the model parameters Θ, namely:

$$P(A, B, C \mid \Theta) \tag{6.1}$$

In contrast, when the auto-regressive approach is presented with the same task, what it needs to learn is the chain of conditional probability distributions defined by:

$$P(A \mid \Theta) \cdot P(B \mid A, \Theta) \cdot P(C \mid A, B, \Theta) \tag{6.2}$$

In the second case, the Transformer is asked to produce one element of the collection at a time, by exploiting information about the previous predictions. This is similar to machine translation, in which the Transformer outputs one word, conditioned on the previously generated output. Eq. (6.1) and Eq. (6.2) are equal by definition of the general product rule

of probability, and Transformers are capable of modelling both. However, using Eq. (6.2) with Transformers imposes an order even when there might not be a natural order present.

In this work, we study the difference between auto-regressive and parallel Transformers on polygonal shape prediction viewed as a collection of points, including individual points, lines, gates and polygons, as shown in Figure 6.1. Our contributions are: (1) We test auto-regressive and parallel models on four collection prediction datasets and one sequential dataset to provide the reader with a full picture of the advantages and disadvantages of both approaches. (2) We show that the conditional decomposition of the collection can be beneficial for Transformers when there is an explicit order in the elements of a set, such as predicting a collection of points on a line. (3) we show empirically that the conditional decomposition benefits from a particular order than others on a polygon dataset.

6.2. RELATED WORK

Polygonal shape detection: The work in [2] views satellite image mapping as a polygon prediction task and makes use of a convolutional network to output the polygon vertices directly. The work in [4] first detects bounding boxes of buildings, and then uses recurrent networks to extract vectorized building footprints. The work in [1] proposes to detect frame field for constructing building topology via fully convolutional networks and is able to precisely segment aerial images. Different from these works, we study the effect of self-attentions or Transformers in polygonal shape prediction.

Object detection as set prediction. DETR [7] first presents object detection as a set prediction task, and removes the need for non-maximal suppression on a large amounts of *anchor boxes*. Similar to DETR, we also consider polygonal shapes in an image as a set. Different from DETR which only output bounding boxes, we produce vectorized polygons as a collection of vertices.

Transformers. Transformers were originally introduced by [13] as a novel auto-regressive, sequence-to-sequence model, and gained popularity thanks to their ability to dispense entirely with recurrence and support parallel processing of sequences. Their stunning results on machine translation and other language tasks [20–23] have recently shed a light on employing Transformers for computer vision tasks, such as image recognition [24], object detection [7], segmentation [25], set prediction [12, 26] and other visual tasks [27–30]. Inspired by these works, we study both auto-regressive and parallel Transformers on polygonal shape prediction.

Transformers for set prediction. Deep Sets [31] has proven that transforming all elements of an input set into some latent representations and then combining them through a permutation invariant function is a universal approximator of any set function. The work in [26] shows that the attention layer of Transformers can also be viewed as a generalization of the sum operation of Deep Sets and is therefore also a universal approximator of set functions. Moreover, Lee *et al.* [12] proposes an attention-based permutation-invariant framework which demonstrates superior results on set-structured data. Inspired by these works, we also study Transformers for set prediction task, but on a particular type of set data - polygonal shapes.

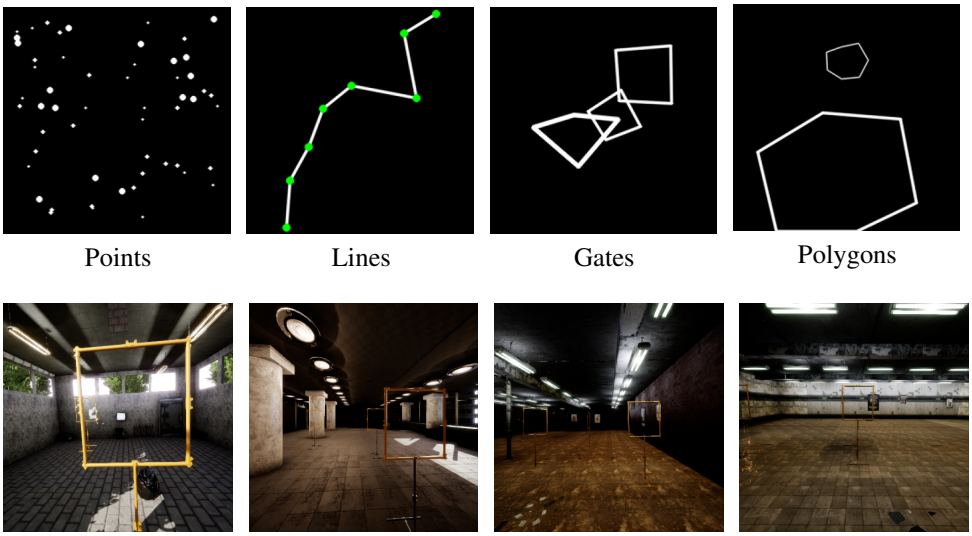


Figure 6.1: **Examples of point collections used in our paper.**

6.3. DATASETS

To illustrate our setting, we first show examples of our datasets in Figure 6.1. We use 4 toy datasets and 1 synthetic dataset. The images in the toy datasets are manually generated during training and if not specified otherwise, we generate 3 million images for training and 10,000 images for testing.

The **point** dataset contains images with n white points randomly distributed over the image space. Each point's size is uniformly sampled from three possible values. The task is to predict the x, y coordinates of all points in any order. This dataset is an instance of a pure set prediction problem, with points representing the set elements.

The **line** dataset contains images picturing a single white line composed by 7 segments. A green point of fixed size is placed on top of each end of a segment. The line is generated by first drawing a straight line going from the bottom left corner of the image to the top right corner, and then randomly shifting 8 equally distributed points perpendicularly to the line direction by $r \in [-15\%, 15\%]$ with respect to the image size. The task is to predict the x, y coordinates of the 8 points following the line order, starting from the end on the bottom left of the image. The set elements are represented in this case by the green points, and this dataset is an example of a set prediction task where there exists an explicit order in which we need to predict its elements.

The **gate** dataset contains images with n convex polygons of 4 corners, called *gates*. Each gate is generated by defining 4 equally spaced points on a circumference of random radius $r \in [5\%, 40\%]$ with respect to the image size. Each point is then shifted randomly in the direction of the radius and the four points are finally connected to define the gate. All four edges of the gates are of the same thickness, uniformly sampled out of 3 possible choices.

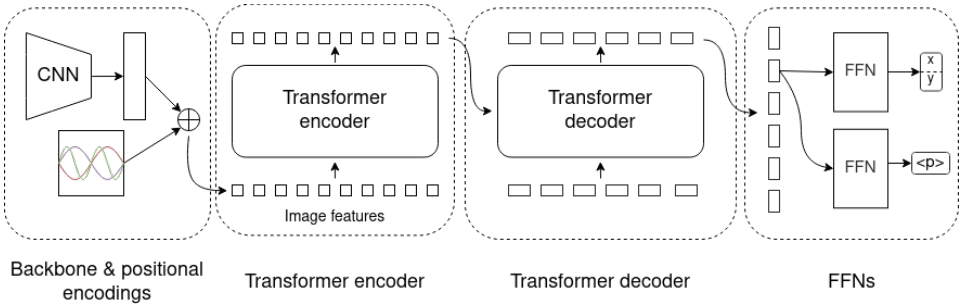


Figure 6.2: **The auto-regressive model for predicting polygons.** This model exploits the *sentences of tokens* idea [13] which generates tokens auto-regressively. There are only four possible tokens in its vocabulary: *start*, *point*, *end-of-polygon* and *end*. The feed-forward network (FFN) at the end of the pipeline produces a class label and a pair of coordinates (x, y).

The **synthetic gate** dataset [32] contains realistic synthetic images generated with a graphical engine, which simulate the flight of a drone in different environments containing empty wireframe objects (EWFOs or *gates*). This dataset is a simulation of the images that a UAV would face in the IROS 2018 Autonomous Drone Race [5]. The training set contains 26,000 images from two different scenes, with light coming from outside and inside the rooms. The test set contains 3,000 images from two additional scenes with light coming only from artificial sources placed inside the rooms. For all scenes, different walls and pavement textures are used, as well as different artificial shapes and light intensities for internal lamps. All images contain 1 to 4 gates. For both synthetic and toy gate datasets, the task is to predict the position of the four corners in the image for each gate. These two datasets represent a set prediction scenario closer to real-life problems, where the complexity of the single set element prediction (the gate) is greater than a simple point prediction. The synthetic gates dataset increases the complexity even further by picturing gates with different backgrounds and different lighting situations.

Finally, the **polygon** dataset contains images with n polygons of $m \in [3,7]$ corners, generated using the same technique as the gate dataset. As multiple non-convex polygons can be represented by the same set of points, the task is to predict the m corners of a polygon in clockwise order. Any starting point is accepted, and distinct polygons can be predicted in any order. This dataset represents the hardest set prediction task in which we require our models to predict a *set of ordered sets*.

For all datasets, n is a parameter varied for different experimental settings. All samples in the toy datasets are RGB images of size 256x256, while images from the synthetic gate dataset are of size 400x400. The coordinates of the labels are represented as the relative height/width to the image size with values in $[0,1]$.

6.4. MODELS

All of the models that we implemented and tested are derived from DETR [7], an end-to-end Transformer which achieves competitive results against Faster R-CNN [33] on object

detection. It takes advantage of a CNN backbone and parallel encoding-decoding Transformers to solve object detection as a set prediction task. In short, DETR is composed of four modules: a CNN backbone, a Transformer Encoder, a Transformer Decoder, and a feed-forward network (FFN). An example of our models is shown in Figure 6.2. It pictures the auto-regressive variant used for the polygon toy setting dataset.

DETR takes as input an RGB image and extracts a high-level feature map of shape $C \times H \times W$, where C is the size of number of channels, and H and W are the spatial dimensions. The feature map is supplemented with fixed positional embeddings before the Transformer encoder. The Transformer encoder is a stack of 6 self-attention mechanisms, each of which consists of a standard self-attention layer followed by a feed forward network, a residual connection and layer normalization [7]. All of the models we tested are identical up to this stage of the architecture, but differ in the subsequent modules.

6.4.1. PARALLEL MODELS

The design of parallel models is identical to DETR [7]. The N learned *object queries* are converted *in parallel* into N output embeddings of size 256 by the Transformer decoder, which are subsequently fed into a simple feed forward network (FFN) for classification and position regression.

Depending on the task, we modify the dimensionality of the FFN for position regression accordingly. For example, on the point and the line datasets, the output is a vector of size 2 representing the (x, y) coordinates of a point normalized by the image width and height, while the output on the gate dataset is an 8-element vector indicating the four vertices of a gate in clock-wise order. The class labels for all datasets are the same, namely the *object* class and the *no-object* class.

On the polygon dataset, the FFN is replaced by a simple multi-layer Elman RNN [34], since the output dimension is also a variable. The RNN model takes as input the embeddings from the *object* class and generates a sequence of points one by one. The RNN is possibly the minimal modification we could make to the architecture to work with polygons and have the smallest impact on the model's properties and our experiments.

6.4.2. AUTO-REGRESSIVE MODELS

The auto-regressive models diverge from the parallel ones on all set prediction tasks, because they work with *sentences of tokens* that are generated one by one, by conditioning the next prediction on the previous ones. In particular, each token is a vector of dimension 256 that represents an element of the set, or acts as a special representation. Special tokens can be:

- *start* or **S**: This token is at the beginning of all sentences and defines the starts of the computation.
- *end* or **E**: This token is at the end of all sentences and terminates the computation.
- *end-of-polygon* or **EOP**: This token separates polygons from polygons inside the same image. It acts similarly to the *period* token used in machine translation to separate words belonging to different sentences.

Depending on the task, object tokens can be:

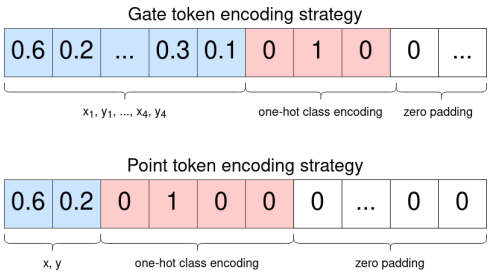


Figure 6.3: **Token embeddings for predicting gates and polygons**. The first $2n$ elements (in blue) of the vector represent the (x, y) coordinates of the n vertices/points, normalized by the image width and height. The following elements (in red) represent the class label in one-hot format. For gate detection, we have three red blocks because we have two special tokens *start* and *end*, and an object token *gate*. For polygon detection we have 4 red blocks because all possible tokens are *start*, *end*, *point* and *end-of-polygon*, as explained in 6.4.2. The vector is padded with zeros to reach the required dimensionality of 256.

- *point* or **P**: On the point dataset and the line dataset, each point is a 2-element vector representing the position of the (x, y) coordinate normalized by the image width and height.
- *gate* or **G**: On both toy and synthetic gate datasets, a gate is represented as a vector of size 8, which defines the normalized coordinates (x, y) of all 4 vertices.

To batch sequences of different lengths together, we pad sentences with the *end* token up to the same length. For example, given a batched sequences where the maximal length is 8, we pad the shorter sequence with 4 visible points only with two extra **E**:

S, **P**, **P**, **P**, **P**, **E**, **E**, **E**

We use the same decoder as the parallel models, but adopt the *masking* technique for parallel training, which prevents each token from attending to subsequent positions [13]. The token embedding is a fixed length vector of size 256 as shown in Figure 6.3.

The auto-regressive approach predicts all polygons in a certain order. If not specified otherwise, we always predict objects going from left to right in the image, splitting ties with top-to-bottom order. We sort polygons and gates accordingly by their centers, computed as the average of their vertices. Particularly, for the polygon dataset, we also impose an order on points such that the polygon is clock-wisely defined by the points, otherwise the same set of the points may result in several polygons of different shapes.

6.5. EXPERIMENTS

First, we validate that the auto-regressive approach is preferable on the line dataset where there is an explicit natural order in the predictions, and that the parallel solution is better on the point dataset for pure set prediction tasks without orders. We then compare the two strategies on more challenging gate datasets where the element is no longer a single point and where data is scarce. Finally, we explore deeper into the auto-regressive solution to study upon the importance of the prediction order and the order of the conditional variables.

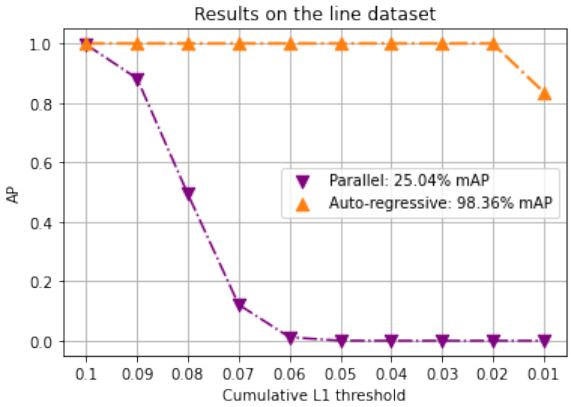


Figure 6.4: **Results on the line dataset.** The task is to predict a series of ordered points. The auto-regressive model produces approximately perfect prediction, while the parallel model fails.

On toy datasets, we train all models with 3,000,000 images. On the real gate datasets with 26,000 images, we train all models for 300 epochs. We apply multiple data augmentation techniques, including horizontal flip, vertical flip, hue shift, Gaussian noising.

We train all models on a single NVIDIA GeForce RTX 2080 Ti GPU with AdamW [35], and set the Transformer's learning rate to 10^{-4}, the backbone's learning rate to 10^{-5}, and the weight decay to 10^{-4}. The learning rate is dropped by a factor of 10 after 200 epochs, or after 2,000,000 images for the toy settings. Mask R-CNN [36] is also trained for 300 epochs with learning rate 5×10^{-3}, which is decreased by 10 after 200 epochs.

6.5.1. EVALUATION

The evaluation metric on the gate and polygon datasets is mean Average Precision [7], averaged over different IoU thresholds ($[0.50, 0.55, \ldots, 0.95]$). On the point and line datasets, we also evaluate mean Average Precision, but averaged over different point-to-point L_1 distance thresholds ($[0.10, 0.09, \ldots, 0.01]$), computed directly on the relative coordinates.

6.5.2. LINE AND POINT DETECTION

With this experiment we show that the conditional decomposition of collections by auto-regressive Transformers is beneficial when the elements in a collection adhere to a natural order. Results on the line dataset are shown in figure Figure 6.4. For this task, a line prediction is considered as a false positive if the sum of all L_1 point distances is greater than the given threshold. The experiment shows that in this setting the auto-regressive solution is much more precise than the parallel counterpart as it is able to achieve perfect mAP up to a threshold of 0.02.

On the other hand, the experiments on the point dataset study the behaviour of parallel and auto-regressive models on a pure set prediction task. The models are now expected to predict the n points in any order. The parallel Transformers is order-insensitive because of the Hungarian matching, but the auto-regressive Transformers are trained by always

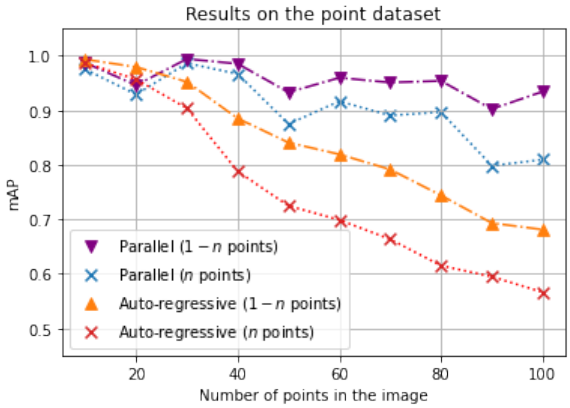


Figure 6.5: **Results on the point dataset.** The auto-regressive model shows marginally better performance than the parallel one on collections of limited cardinality, but its performance is lacking when the cardinality grows. The legends with *n points* represent results on images with exactly n points while the others indicate that the number of points in an image varies from 1 to n.

feeding points in the left-to-right order. Results on this dataset are shown in Figure 6.5. In this experiment, we train all models on images with 1 to n points. We present test results on images with 1 to n points, as well as results on images with exactly n points. Results show that on collection prediction tasks, the auto-regressive approach is effective when the cardinality is low, but quickly deteriorates as the cardinality increases.

The two experiments on lines and points prove that the presence of a natural order in the task is indeed an important discriminative factor towards the performance of the parallel and auto-regressive models in collection prediction. Moreover, we show the advantage of the auto-regressive approach in low-cardinality prediction tasks. The parallel ones show significant advantage in predicting high-cardinality collections but at the cost of using redundant object queries (100 object queries in this experiment).

6.5.3. GATE DETECTION

The following experiments verify whether the observations on the line and point datasets generalize in complicated scenarios where the collection element is a gate of four vertices. First, we evaluate the performances of auto-regressive approach and its parallel counterpart on the toy gate dataset, as shown in Figure 6.6. There are two parallel models in comparison: one with oversampling where we use 30 object queries which is several times more than the total number of gates, and the other one without oversampling where we use as many object queries as the maximal number of objects in the images. The parallel model with oversampling outperforms the one without oversampling substantially, validating the need for the parallel models to oversample the cardinality. Moreover, we show once more the auto-regressive approach performs comparably to the parallel one on low-cardinality collections, but suffers from the growing cardinality.

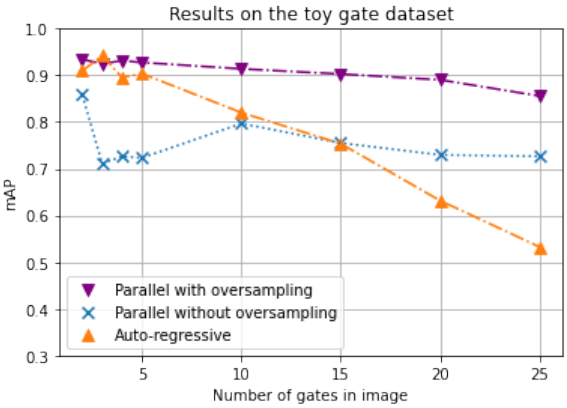


Figure 6.6: **Results on the gate dataset.** Each point of the plot is an experiment in which the model is trained and tested on images with 1 to n gates. They show the necessity of oversampling for parallel Transformers. The performance of auto-regressive models deteriorates when collection cardinality increases.

In Figure 6.7, we provide our findings on the generalization capabilities of these models, where the number of gates in test images is up to twice more than the number during training. The auto-regressive model struggles at detecting more gates, while the parallel one only shows minor performance decrease and achieves an mAP of 0.8 even on images with twice the amount of gates. We believe this behavior is a result of oversampling: each of the 30 object queries is assigned to at least one gate during training, which implicitly tells the model that there could be more objects than the ones it sees in each image.

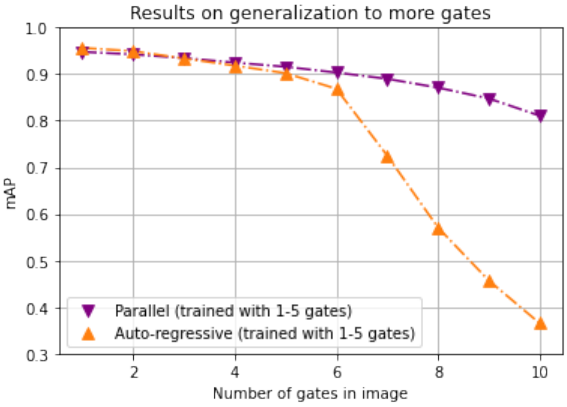


Figure 6.7: **Generalization capability of the different models.** We test both parallel and auto-regressive on images with twice as many gates as the training images. The parallel model is able to detect many more gates than the auto-regressive one. We believe oversampling attributes to better generalization.

Finally, we numerically compare the general performances of all models on both toy and synthetic gate datasets with images containing 1 to 6 gates. We choose Mask R-CNN as the baseline. Table 6.1 shows parallel Transformer outperforms Mask R-CNN by 5 % mAP on the toy dataset and by 4 % mAP on the synthetic gate dataset. The auto-regressive approach is not able to achieve comparable results as there is no natural order in this task. However, the auto-regressive model is approximately two times faster than the parallel model as it does not require Hungarian algorithm or oversampling.

Models	Toy gates	Training time	Synthetic gates	Training time
Mask R-CNN	90.67%	23.6	61.30%	13.85
Parallel	**95.45%**	2.2	**65.00%**	3.85
Auto-regressive	87.67%	**1.5**	59.03%	**1.85**

Table 6.1: **Numerical results on gate datasets**. We evaluate the overall performance in mAP averaged over 10 IoU thresholds [0.50, 0.55, ..., 0.95]. The parallel model outperforms Mask R-CNN. The performance of auto-regressive model is lacking due to the absence of natural order in this task. The unit of training time is minutes per 10K images.

6.5.4. POLYGON DETECTION

Results on the polygon toy setting dataset are shown in table Table 6.2. This experiment explores an hybrid scenario in which the collection elements we are trying to predict are also collections with an imposed order, because a collection of points can represent multiple polygons if the order of its points is not given. The auto-regressive approach is the special one described at the end of section 6.4.2, which predicts a polygon as a sequence of points followed by the *end-of-polygon* token, and it outperforms the parallel model substantially by over 20 % in terms of mAP. When using the auto-regressive solution on polygons, we are predicting each polygon individually by predicting its vertices and then an end-of-polygon token. This means that we are splitting a problem of directly predicting all points of all polygons into multiple sub-problems of predicting each polygon and then deciding when all points have been predicted. We speculate that the advantage of the auto-regressive model could be a direct consequence of the conditional decomposition of the joint probability, as imposing the conditional order on these models can serve as a strong inductive bias by reducing the search space of the model. This behaviour might not show up on the previous experiments because gates, points and lines are simple collection elements which do not enlarge the search space enough. As polygon detection is representative of many common computer vision tasks, we leave further exploration of this approach as future work.

6.5.5. ORDERS OF CONDITIONAL DECOMPOSITION

We conclude our experiments by providing insights on the importance of orders imposed on conditional decomposition of a collection by the auto-regressive Transformers. Moreover, we also study the influence of positional endcodings. We run experiments on the toy gate dataset, synthetic gate and the polygon datasets multiple times, by using two different object orderings and by adding or removing the positional encodings of the attention layer.

Models	Polygons dataset	Training time [mins/10k images]
Parallel	53.55%	5.1
Auto-regressive	**76.41%**	**2.15**

Table 6.2: **Numerical results on the polygon dataset.** Scores are represented as mAP averaged over 10 IoU thresholds [0.50, 0.55, . . . , 0.95]. The auto-regressive approach outperforms the parallel one by over 20% absolute mAP. We speculate that this could be a direct consequence of conditional decomposition of the collection.

	Positional encodings	No positional encodings
Order based on polygon positions	$-1.90\% \pm 1.07\%$	–
Order based on the size of polygons	$-7.68\% \pm 4.53\%$	$-2.95\% \pm 1.30\%$

Table 6.3: **Ablation studies on position encodings and condition orders.** This tables shows results of adding positional encodings and imposing orders. The top performing model is spatially ordered (left-to-right and top-to-bottom) and has no positional encoding. Our observations are that: (1) changing the imposed order on auto-regressive Transformers has a great impact on the performance as shown in the first column; (2) adding position encodings decreases the performance.

Without the positional encodings, the Transformer decoder is unaware of the index of the previous predictions in the sequence. When predicting a new point, it only knows which points were predicted before, but not their orders. The two orderings are the left-to-right, top-to-bottom order and the small-to-large order in terms of the size of the objects. Experiment results are shown in table Table 6.3. Using the left-to-right, top-to-bottom order and removing the positional encodings lead to the best performance in all experiments. We show the mean absolute difference of other models compared to best one. The result shows that the artificial order *matters* for auto-regressive models. Moreover, we show that fixed positional encodings are not beneficial in this setting, in contrast to the original Transformer architectures. This is expected as different orders of the same words in natural language processing can have different meaning, while this does not matter in our setting, as knowing the location of a point can already prevent our model to predict the same point twice.

6.6. CONCLUSION

We studied two important variants of the Transformer architecture: the parallel one and the auto-regressive one on the polygonal shape prediction task cast as a collection prediction problem. We find that the ordering of objects is a strong factor towards the performance of these models on point, line, gate and polygon datasets. The auto-regressive model benefits collections of small cardinality or on collection prediction problems that can be easily split into multiple easier tasks.

One limitation of this research is that most experiments are conducted on toy datasets only, and it is unclear whether the observations and conclusions on toy datasets generalize on challenging real-world datasets. A task that would be suitable to expand our research is polygonal building segmentation from satellite images, as in [1, 37].

As future work, we find it important to further explore on the polygon detection task as it is a fundamental problem that is representative in computer vision. Testing the effect of fixed or learned token embeddings would also be of interest.

REFERENCES

[1] N. Girard, D. Smirnov, J. Solomon, and Y. Tarabalka, *Polygonal building segmentation by frame field learning,* arXiv preprint arXiv:2004.14875 (2020).

[2] N. Girard and Y. Tarabalka, *End-to-end learning of polygons for remote sensing image classification,* in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium* (IEEE, 2018) pp. 2083–2086.

[3] V. Iglovikov, S. Seferbekov, A. Buslaev, and A. Shvets, *Ternausnetv2: Fully convolutional network for instance segmentation,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2018) pp. 233–237.

[4] Z. Li, J. D. Wegner, and A. Lucchi, *Topological map extraction from overhead images,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 1715–1724.

[5] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, *A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge,* Journal of Field Robotics **35**, 146 (2018).

[6] Y. Zhang, J. Hare, and A. Prügel-Bennett, *Fspool: Learning set representations with featurewise sort pooling,* arXiv preprint arXiv:1906.02795 (2019).

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers,* in *European Conference on Computer Vision* (Springer, 2020) pp. 213–229.

[8] Y. Zhang, J. Hare, and A. Prügel-Bennett, *Deep set prediction networks,* arXiv preprint arXiv:1906.06565 (2019).

[9] Z. Cao, R. Wang, X. Wang, Z. Liu, and X. Zhu, *Improving human pose estimation with self-attention generative adversarial networks,* in *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (IEEE, 2019) pp. 567–572.

[10] W. Tang and Y. Wu, *Does learning specific features for related parts help human pose estimation?* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 1107–1116.

[11] S. H. Rezatofighi, V. K. BG, A. Milan, E. Abbasnejad, A. Dick, and I. Reid, *Deepsetnet: Predicting sets with deep neural networks,* in *2017 IEEE International Conference on Computer Vision (ICCV)* (IEEE, 2017) pp. 5257–5266.

[12] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, *Set transformer: A framework for attention-based permutation-invariant neural networks,* in *International Conference on Machine Learning* (PMLR, 2019) pp. 3744–3753.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need,* arXiv preprint arXiv:1706.03762 (2017).

[14] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate,* arXiv preprint arXiv:1409.0473 (2014).

[15] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, *Imputer: Sequence modelling via imputation and dynamic programming,* in *International Conference on Machine Learning* (PMLR, 2020) pp. 1403–1413.

[16] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, *Mask-predict: Parallel decoding of conditional masked language models,* arXiv preprint arXiv:1904.09324 (2019).

[17] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, *Non-autoregressive neural machine translation,* arXiv preprint arXiv:1711.02281 (2017).

[18] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, *Deformable detr: Deformable transformers for end-to-end object detection,* arXiv preprint arXiv:2010.04159 (2020).

[19] H. W. Kuhn, *The hungarian method for the assignment problem,* Naval Research Logistics (NRL) **52**, 7 (2005).

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, *Language models are few-shot learners,* arXiv preprint arXiv:2005.14165 (2020).

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding,* arXiv preprint arXiv:1810.04805 (2018).

[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach,* arXiv preprint arXiv:1907.11692 (2019).

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training,* (2018).

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

[25] L. Ye, M. Rochan, Z. Liu, and Y. Wang, *Cross-modal self-attention network for referring image segmentation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 10502–10511.

[26] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne, *On the limitations of representing functions on sets,* in *International Conference on Machine Learning* (PMLR, 2019) pp. 6487–6494.

[27] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, *Pre-trained image processing transformer,* arXiv preprint arXiv:2012.00364 (2020).

[28] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, *Video action transformer network,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 244–253.

[29] C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid, *Videobert: A joint model for video and language representation learning,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 7464–7473.

[30] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, *Learning texture transformer network for image super-resolution,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020) pp. 5791–5800.

[31] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, *Deep sets,* arXiv preprint arXiv:1703.06114 (2017).

[32] P. Dürnay, *Detecting empty wireframe objects on micro-air vehicles: Applied for gate detection in autonomous drone racing,* (2018).

[33] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks,* arXiv preprint arXiv:1506.01497 (2015).

[34] A. Sherstinsky, *Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network,* Physica D: Nonlinear Phenomena **404**, 132306 (2020).

[35] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization,* arXiv preprint arXiv:1711.05101 (2017).

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn,* in *Proceedings of the IEEE international conference on computer vision* (2017) pp. 2961–2969.

[37] S. P. Mohanty, J. Czakon, K. A. Kaczmarek, A. Pyskir, P. Tarasiewicz, S. Kunwar, J. Rohrbach, D. Luo, M. Prasad, S. Fleer, *et al.*, *Deep learning for understanding satellite imagery: An experimental survey,* Frontiers in Artificial Intelligence **3** (2020).

7

DISCUSSION

7.1. CONCLUSION

This thesis explores adding priors into neural networks for detecting geometric primitives from RGB images, so that the inductive knowledge does not need to be learned from massive data. We have made advances by injecting a variety of geometric priors as end-to-end trainable neural network modules. In particular, we studied the added value of adding geometric priors on five tasks: (1) wireframe parsing, (2) lane detection, (3) vanishing point detection, (4) 3D mirror symmetry detection and (5) polygonal object detection. Extensive experiments show the advantage of these priors in improving overall performance, especially in a small-data regime.

The geometric priors studied in this thesis include Hough Transform, Gaussian sphere mapping and 3D mirror symmetry. Despite of being task-specific, all the priors share a common strategy - global voting in the parameter space. Taking the Hough Transform as an example where each line is parameterized as a bin, we first calculate all possible lines that a point might belong to, and then increment the bins accordingly by the value of the given point. After enumerating all points, we obtain the entire accumulator where the intensity indicates the number of points on the given line. Therefore, we can identify lines by searching for high-intensity bins in a local neighborhood. The suitability of the Hough Transform stems from its ability to aggregate local features into a global parameterization, as we collect evidence for the presence of lines from all points. This strategy is often referred to as Hough voting in a variety of applications, such as point cloud processing [1, 2], image matching [3] and object detection [4, 5].

Similarly, one might also consider the Gaussian sphere mapping as another variant of Hough voting, where the parameter space is a unit hemisphere, represented by azimuth and elevation, since vanishing points are directions of parallel lines in the 3D world. Concretely speaking, we start with the interpretation plane formed by a line and the camera origin. The interpretation plane intersects the unit sphere with a great circle, which indicates all possible vanishing points that the given line might point at. Since vanishing points are the intersection of several lines, we can simply identify vanishing points by examining the

overlap of multiple great circles. Higher intensity implies a larger chance for the presence of vanishing points. Although the parameterization differs from the Hough Transform, the Gaussian sphere mapping is also an instance of Hough voting, where each line votes on a unit hemisphere.

The Hough voting has a unique property - globality. For instance, an individual bin in the Hough Transform “observes" an entire straight line, which is often a sequence of spatially adjacent pixels in images. In the Gaussian sphere mapping, a spherical point “perceives" all possible lines that vanish along the given direction. Alternatively, we can interpret the Hough voting as a layer whose receptive field is the entire primitive. In contrast, the commonly used filtering (or convolution) works with local patches only. Thus, it essential to stack multiple layers to enlarge the receptive field such that a network “sees" the entire image. However, this introduces extra computation inevitably. In light of the distinction between “globality" and “locality", we proposed to incorporate the Hough Transform prior into a neural network. We not only took the advantage of convolutional networks in local feature learning, but also inherited the strength of Hough voting in global parameterization. The benefits of incorporating Hough voting are consequently two folds. On one hand, we no longer depend on deeper neural networks to ensure a sufficiently large receptive filed, thus reducing the number of parameters. On the other hand, specialized knowledge of the geometric primitives was explicitly injected into learning without learning from massive data, thus mitigating the demand for labeled data substantially.

We parameterize geometric shapes and then apply filtering on those parameterizations, which allows us to identify geometric structures directly. However, our neural networks failed to deliver competitive results without properly initializing the incorporated priors, as demonstrated in chapter 2. Our speculation was that adding knowledge constrains the solution space and thus requires specialized filters. We drew inspirations from the (Inverse) Radon Transform, where ramp filters are used for high-quality reconstruction [6], and initialized the filters in the Hough domain in a similar way. The intuition comes from the fact that the Hough Transform is equivalent to a discrete variant of the Radon Transform [7]. This choice had dramatic impact on learning and significantly improved the performance. Therefore, we conclude that specialized knowledge about initialization is necessary to fully leverage the potential of the added geometric priors.

One major drawback of the Hough voting is quantization. A fine-grained quantization is essential to precisely identify geometric structures. However, this will inevitably increases the memory consumption and inference latency. Moreover, globality is also a non-negligible factor that contributes substantially to computation with the increase of dimensionality, as we exhaustively enumerate over each grid point and calculate its mapping over the entire parametric space. Fortunately, there are certain assumptions we can exploit to reduce unnecessary computation, e.g., the Manhattan world assumption in vanishing point detection, the regularity of geometric structures, and the sparsity in humanly constructed scenes. These assumptions allow us to calculate the parameterization for only a fraction of grid points, thus dramatically reducing the computation cost. Another solution to remove the discrete quantization is analytic parameterization, such as implicit functions where multi-layer perceptrons are adopted to represent continuous signals [8–12].

7.2. FUTURE WORK

While this thesis made progress in several tasks, we propose a number of research topics as future work where our findings can be useful.

Geometric primitive detection by pretraining on synthetic data. Although incorporating geometric priors improves data-efficiency, the proposed approach still requires hundreds of labeled images for training. However, most geometric primitives are artificial in nature, and thus we can manually create synthetic datasets for these tasks where all primitives are well-calibrated. Pretraining on these datasets would facilitate neural networks to learn the filters in the parameter space [13]. Furthermore, we can fine-tune the pretrained models with only a handful of annotated real-world images to achieve desired performance.

3D wireframe parsing from single 2D images. Intuitively, we can vectorize Manhattan frames as 3D wireframes, a collection of junctions, line segments and their topology relations [14, 15]. Therefore, it would be intersting whether we can leverage line-related geometric priors to reconstruct the 3D layout of Manhattan frames together with the orthogonality and parallelism assumptions. On one hand, we have demonstrated the advantage of the Hough Transform line priors in chapter 2 for extracting 2D wireframes. On the other hand, our work in chapter 4 has illustrated the usage of Gaussian sphere mapping for vanishing point detection, from which we can identify the 3D orientation of associated line segments [16]. One promising idea is to combine the Hough Transform line priors and the Gaussian sphere mapping into a unified framework for 3D Manhattan frame reconstruction.

Parametric structure detection from point clouds. One popular representation of 3D data is point clouds. The works in [17, 18] have shown impressive results in detecting parametric curves from point clouds, but at the cost of exhaustive search. In contrast, the work in [2] proposes a compute-efficient Hough voting strategy for detecting 3D primitives from point clouds, such as cylinders, spheres and cones. One promising future work is to further explore the Hough voting scheme in processing large-scale point clouds which are typically incomplete, noisy, and unorganized [19].

Deformable geometric primitives. Objects vary in scale, pose and gesture. Humans learn this knowledge arithmetically from education or manually from real-world experience. One might wonder how to deform a geometric structure with neural networks. The priors studied in this thesis provide useful tools for deformation as a primitive deforms in accordance with its parameterization. Therefore, the task is equivalent to estimating the optimal values of certain parameters that control the deformation. One nice property of this approach is that these parameterizations are physically meaningful as we can easily visualize the effect of parameter changing. What is more, the deformation of geometric structures is purely artificial and thus we can use graphics engines for generating ample synthetic data to simulate the physical world.

Scene compression. Online streaming is popular in digital entertainment, TV broadcasting, and remote meetings. However, transferring data is challenging due to the amount of information and real-time latency. One possible solution to compress the data is to leverage the repetition and coherency of the geometric world, such as windows on a wall, tables in a classroom, and buildings in a residential block [20]. We can represent recurring objects with parameterized templates, but at different positions. Moreover, the parameters of various templates can be precisely estimated from the Hough voting. This brings benefit to

memory storage and computational efficiency as we replace duplicate representations with shared parameterization.

Geometric structure decomposition. Artificial objects are often a composition of multiple geometry primitives. A straightforward extension of our work is to disentangle seemingly complicated objects into simple primitives. A minimal example is polygons, which can be decomposed into a *collection of ordered vertices*. We have provided evidence that auto-regressive transformers can be a feasible solution to learn polygonal representations on toy datasets in chapter 6. Thus it would be interesting to generalize our observations to real-world tasks, such as polygonal building segmentation and indoor floorplan estimation [21]. This will shed light on other high-level tasks, such as inferring semantically consistent part arrangements for object instances [22] and reasoning part-whole hierarchies [23].

7.3. FINAL REMARK

We have shown that detecting geometric primitives is a demanding task, where the challenge comes from acquiring large amounts of labeled data, especially in the deep learning era. With this thesis, we made one step forward by fusing classic geometric priors with modern deep learning techniques, and demonstrated the potential of visual inductive priors in improving data-efficiency. Nevertheless, there is still a long journey ahead to fully exploit the capacity of deep neural networks in understanding the artificial world.

REFERENCES

[1] C. R. Qi, O. Litany, K. He, and L. J. Guibas, *Deep hough voting for 3d object detection in point clouds,* in *Proceedings of the IEEE International Conference on Computer Vision* (2019) pp. 9277–9286.

[2] C. Sommer, Y. Sun, E. Bylow, and D. Cremers, *Primitect: Fast continuous hough voting for primitive detection,* in *IEEE International Conference on Robotics and Automation (ICRA)* (2020).

[3] J. Min and M. Cho, *Convolutional hough matching networks,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 2940–2950.

[4] B. Leibe, A. Leonardis, and B. Schiele, *Combined object categorization and segmentation with an implicit shape model,* in *Workshop on statistical learning in computer vision, ECCV*, Vol. 2 (2004) p. 7.

[5] N. Samet, S. Hicsonmez, and E. Akbas, *Houghnet: Integrating near and long-range evidence for bottom-up object detection,* in *European Conference on Computer Vision* (Springer, 2020) pp. 406–423.

[6] M. Magnusson, *Linogram and Other Direct Fourier Methods for Tomographic Reconstruction*, Linköping studies in science and technology: Dissertations (Department of Mechanical Engineering, Linköping University, 1993).

[7] M. van Ginkel, C. L. Hendriks, and L. J. van Vliet, *A short introduction to the radon and hough transforms and how they relate to each other,* Delft University of Technology (2004).

[8] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, *Occupancy networks: Learning 3d reconstruction in function space,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 4460–4470.

[9] Z. Chen and H. Zhang, *Learning implicit fields for generative shape modeling,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 5939–5948.

[10] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, *Deepsdf: Learning continuous signed distance functions for shape representation,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019) pp. 165–174.

[11] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions,* Advances in Neural Information Processing Systems **33** (2020).

[12] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis,* in *European conference on computer vision* (Springer, 2020) pp. 405–421.

[13] R. Pautrat, L. Juan-Ting, V. Larsson, M. R. Oswald, and M. Pollefeys, *Sold²: Self-supervised occlusion-aware line description and detection,* in *Computer Vision and Pattern Recognition (CVPR)* (2021).

[14] Y. Zhou, H. Qi, Y. Zhai, Q. Sun, Z. Chen, L.-Y. Wei, and Y. Ma, *Learning to reconstruct 3d manhattan wireframes from a single image,* in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) pp. 7698–7707.

[15] Y. Zhou, J. Huang, X. Dai, L. Luo, Z. Chen, and Y. Ma, *Holicity: A city-scale data platform for learning holistic 3d structures,* arXiv preprint arXiv:2008.03286 (2020).

[16] Y. Lin, S. L. Pintea, R. Wiersma, K. Hildebrandt, E. Eisemann, and J. C. van Gemert, *Geometric priors for deep vanishing point detection,* in *Technical report* (2021).

[17] Y. Liu, S. D'Aronco, K. Schindler, and J. D. Wegner, *Pc2wf: 3d wireframe reconstruction from raw point clouds,* in *International Conference on Learning Representations* (2021).

[18] X. Wang, Y. Xu, K. Xu, A. Tagliasacchi, B. Zhou, A. Mahdavi-Amiri, and H. Zhang, *Pie-net: Parametric inference of point cloud edges,* arXiv preprint arXiv:2007.04883 (2020).

[19] V. Ganapathi-Subramanian, O. Diamanti, S. Pirk, C. Tang, M. Niessner, and L. Guibas, *Parsing geometry using structure-aware shape templates,* in *2018 International Conference on 3D Vision (3DV)* (IEEE, 2018) pp. 672–681.

[20] B. Angles, *Geometric modeling with primitives*, Ph.D. thesis, Université Paul Sabatier-Toulouse III (2019).

[21] N. Girard, D. Smirnov, J. Solomon, and Y. Tarabalka, *Polygonal building extraction by frame field learning,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 5891–5900.

[22] D. Paschalidou, A. Katharopoulos, A. Geiger, and S. Fidler, *Neural parts: Learning expressive 3d shape abstractions with invertible neural networks,* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021) pp. 3204–3215.

[23] G. Hinton, *How to represent part-whole hierarchies in a neural network,* arXiv preprint arXiv:2102.12627 (2021).

SUMMARY

The humanly constructed world is well-organized in space. A prominent feature of this artificial world is the presence of repetitive structures and coherent patterns, such as lines, junctions, wireframes of a building, and footprints of a city. These structures and patterns facilitate visual scene understanding by providing abundant geometry information. Humans can easily recognize diverse geometric structures. However, we wonder how to instruct an autonomous agent to interpret the visual world as we do?

There has been great interest in automatic understanding of the geometric world from images in the past few decades. Conventional approaches first detect hand-crafted edge features and then group them into parametric shapes such as lines. Recently, this strategy has been gradually replaced by deep neural networks because learning features from large annotated datasets gives a richer representation compared to hand-designed features. Although the progress is inspiring, there are still several concerns on deploying neural networks in the real-world. A primary concern is the availability of massive labeled data, as the performance of deep networks deteriorates substantially when training data is scarce. This thesis introduces novel strategies to enhance the performance of neural networks in a small data regime, by adding geometric priors into learning.

We start with the Hough Transform, a well-known prior for straight lines, and offer a principled way to add this prior into neural networks for data efficient end-to-end learning. On the wireframe parsing task, our model advances the state-of-the-art substantially on various subsets with much less training data. Subsequently, we extend the Hough Transform line priors to semi-supervised lane detection, only requiring a small amount of labeled data, and show that this approach improves the overall performance by leveraging a massive amount of unlabeled data.

We explore a second geometric prior, the Gaussian sphere mapping, for vanishing point detection. We present an end-to-end framework for detecting multiple non-orthogonal vanishing points without relying on large quantities of training samples. Moreover, the proposed model exhibits consistent performance across multiple datasets without fine-tuning, thus demonstrating the effectiveness of geometric priors in tackling data variation.

Next, we study detecting 3D mirror symmetry from single-view images. We explicitly incorporate 3D mirror geometry into identifying symmetry planes. To reduce the computational footprint, we design multi-stage spherical convolutions to hierarchically pinpoint the optimal plane in the parameter space. Our model not only improves overall performance but also reduces the inference latency substantially.

Finally, we explore the possibility of detecting polygonal shapes from images by using transformers. We provide a full picture of the strength and weakness of the auto-regressive and parallel transformers on detecting polygons viewed as *collections of points*. we demonstrate on a toy dataset that the auto-regressive transformers can be a reasonable option for learning polygonal representations from real-world images.

Taken together, with this thesis we show that incorporating geometric priors into modern deep learning allows reducing the need for expensive, manually annotated data.

SAMENVATTING

De door de mens gemaakte wereld bestaat uit repetitieve structuren en coherente elementen, zoals lijnen en hun verbindingspunten, zoals te zien in raamwerken van gebouwen. Deze structuren en patronen maken visueel begrip van scènes mogelijk door informatie over de geometrie te verschaffen. Mensen kunnen allerlei geometrische structuren gemakkelijk herkennen door middel van ervaringen uit de echte wereld. We vragen ons in deze thesis af of een autonome agent de visuele wereld zou interpreteren zoals mensen.

De afgelopen decennia is er veel belangstelling geweest voor het begrijpen van de geometrische wereld vanuit beelden. Conventionele benaderingen detecteren eerst handgemaakte randkenmerken en groeperen ze vervolgens in parametrische vormen zoals lijnen. Onlangs is deze strategie geleidelijk vervangen door diepe neurale netwerken omdat het leren van functies uit grote geannoteerde datasets een rijkere weergave geeft in vergelijking met met de hand ontworpen functies. Hoewel de vooruitgang inspirerend is, zijn er nog steeds verschillende zorgen over de implementatie van neurale netwerken in de echte wereld. Een eerste zorg is de beschikbaarheid van massieve gelabelde gegevens, aangezien de prestaties van diepe netwerken aanzienlijk verslechteren wanneer trainingsgegevens schaars zijn. Dit proefschrift introduceert nieuwe strategieën om de prestaties van neurale netwerken in een klein dataregime te verbeteren, door geometrische prioriteiten toe te voegen aan het leren.

We beginnen met de Hough Transform, een bekende prior voor rechte lijnen, en we geven een principiële manier om deze prior toe te voegen aan neurale netwerken voor data-efficiënt end-to-end leren. Wat betreft het vinden van gebouw raamwerken, verbetert ons model de state-of-the-art aanzienlijk op verschillende subsets met veel minder trainingsgegevens. Vervolgens breiden we de Hough Transform-lijn uit tot semi-gesuperviseerde rijstrookdetectie. We stellen een nieuwe kostenfunctie voor om rijstroken van grote hoeveelheden ongeziene afbeeldingen te identificeren en de algehele prestaties te verbeteren door gebruik te maken van enorme niet-gelabelde gegevens.

We onderzoeken een tweede geometrische prior, de Gaussische bol projectie, voor detectie van verdwijnpunten. We presenteren een end-to-end netwerk voor het detecteren van meerdere niet-orthogonale verdwijnpunten zonder te moeten vertrouwen op grote hoeveelheden trainingsvoorbeelden. Bovendien vertoont het voorgestelde model consistente prestaties over meerdere datasets zonder fine-tuning, waardoor de effectiviteit van geometrische priors wordt aangetoond bij het aanpakken van datavariatie.

Vervolgens bestuderen we het detecteren van 3D-spiegelsymmetrie van een enkel beeld. We nemen expliciet 3D-spiegelgeometrie op in het identificeren van symmetrievlakken. Om de computationele voetafdruk te verkleinen, ontwerpen we meertraps sferische convoluties om hiërarchisch het optimale vlak in de parameterruimte te lokaliseren. Ons model verbetert niet alleen de algehele prestaties, maar vermindert ook de latentie in het maken van voorspellingen.

Ten slotte onderzoeken we de mogelijkheid om veelhoekige vormen uit afbeeldingen te detecteren met behulp van transformers. We illustreren de sterkte en zwakte van de auto-regressieve en parallelle transformatoren bij het detecteren van polygonen die worden gezien als *verzamelingen* van punten. we demonstreren op een volledig gecontroleerde dataset dat de auto-regressieve transformatoren een redelijke optie kunnen zijn voor het leren van veelhoekige representaties van afbeeldingen uit de echte wereld.

Samengevat hebben we met dit proefschrift aangetoond dat het opnemen van geometrische priors in moderne deep learning-technieken de behoefte aan dure, handmatig geannoteerde gegevens vermindert.

ACKNOWLEDGEMENTS

There is an end to every journey. Here I am, looking back at this wide ride. I was more than lucky to have the support from all people, who inspired, motivated, encouraged and trusted me. I would like to take this opportunity to deliver my sincere gratitude to all of you, without whom I would not have been able to reach the end.

First and foremost, I am greatly indebted to my supervisors: Jan and Silvia. Thank you for carrying me along the adventure, especially in the tough days. The essential values of doing research that you passed on to me, I will never forget. I am also deeply grateful to my promotor, Marcel, for your valuable advice on managing PhD life.

Thank you, my committee members: Prof. dr. D. M. Gavrila, Prof. dr. K. J. Batenburg, Prof. dr. T. Gevers, Prof. dr. R. C. Veltkamp, and Prof. dr. ir. R. L. Lagendijk, for reading this dissertation and participating in my defense.

Next, my sincere gratitude goes to the entire PRB group. Dear CVers, it was fantastic to have you around. You definitely deserve an enormous amount of appreciation. Robert-Jan, the FlexConv man who always takes the initiative to get everything well-organized. Of course! Attila, the day-night mate who is particularly into skiing and tennis. Osman, the man who was either having a tea break or on the way to have a tea break. Ombretta, the cheerful Italian colleague who always brings optimism. Yeshwanth, a genuine fellow who stood with me at the beginning of this journey. Amogh, I was impressed by your dancing steps at parties ;). Ziqi, you always have the best idea of having fun! Yunqiang, the smart mind on binary neural networks. Xin, the "remote" colleague working on video analysis. Marian, the cool pupil/gaze *guy*. Nikolaas, the industry colleague who always holds a microphone during virtual meetings. Burak, thank you for the paper reproduction repository. Giorgio, simply the best on the basketball court. Xiangwei, enjoy the coming PhD life. Andrea, I am proud of your work on investigating transformers. Nergis, the rigorous thinker who always impresses me with her deep insights. Xucong, a young academic with an ambition in human avatars. Thank you for the get-togethers. Seyran, the metric leaner in the architecture faculty. Wenjie, thank you for the advice on career planning through the years.

Dear PRers, your company also made a long journey less lonely. Marco, thank you for the thought-provoking comments, which refresh my research vision. David, the greatest story-teller ever. You fully deserve a spot in Hollywood. Jesse, thank you for the insightful questions and discussions in lab/coffee talks. Tom, I will always remember these entertaining moments we had at the pride parade and the boat party. Alexander, our Mr Muscle, we miss you in the gym. I would also like to thank other PRers for the joyful moments together: Taygun, Skander, Stephan, Arman, Ramin, Rickard, Jin, Yuko, and Taylan.

Dear SPCers, you're an invaluable part of this journey. Hayley, many thanks for sharing your vision. I always appreciate the unique perspective you bring to work. You have taught me (implicitly) more than doing research. Chirag, thank you for being there at borrels, in the office and on the football field. Stephanie, an easy-going officemate to whom I can always talk about life and work. Thank you for the out-of-the-box coffee talks. Jose, the

man working on laughter detection seldom laughs. Ekin, our corridors became silent since you left. I would also like to say thank you to Laura, Yanxia, and Bernd for being great colleagues.

Dear Bioers, I enjoyed great time with you as well. The office wouldn't be the same place without you. Tamim, the only officemate during the pandemic. I truly appreciate your help around the office. Ramin, another officemate known for his kindness and expertise in troubleshooting. Stavros, the bio deep-learner who has the most tricks and ideas on roasting (comedy). Yasin, the only cake I liked was from you! Meng, simply want to say thank you for being there. I would also like to thank the other bio colleagues: Thomas, Joana, Ahmed, Erik, Jasmine, Arlin, Sally, Mo, Mostafa, Lieke, Tom, Soufiane, Chengyao, Stephanie, Colm, Akash, Aysun, Jasper, Paul, Julian, and Erin.

I am going to miss the coffee talks, borrels, retreats, hang-outs, barbecues, football, basketball tournaments, etc. Thank you for all the wonderful moments we have shared. Special thanks also goes to Saskia, Ruud and Bart for the timely support.

I would also like to thank Ruben Wiersma for being a great collaborator. You have opened the door to computer graphics for me. I wish you a pleasant PhD journey.

I should mention Rudy De Lange, for being a great fitness instructor. I'm impressed and inspired by the optimism you have poured into daily work. Thank you!

I enjoyed great moments with all my friends in Delft, especially my neighbors: Qiyao Peng, Kailun Yang, Zhe Li and Yang Xiao. Thank you for the hotpot and board games, and for sharing joys and sorrows. I would also like to acknowledge the support from Nijmegen: Zhengyu Zhao, Wei Xue, and Jie Liu. Thank you all for the joyful times.

Last but not least, no one deserves greater thanks than my parents, who gave me selfless support and unreserved love all long this journey. I can not thank you enough; Still I want to say "thank you".

Yancong Lin
March, 2022
Delft, The Netherlands

CURRICULUM VITÆ

Yancong LIN

11-10-1991 Born in Qixia, Shandong Province, China.

EDUCATION

2010-2014 BSc in Electronic Information Science and Technology
Southwest Jiaotong University, Chengdu, China

2014-2017 MEng in Electronic and Communication Engineering
Tianjin University, Tianjin, China

2017-2021 PhD candidate, Computer Vision Lab
Delft University of Technology, Delft, The Netherlands
Thesis: Data-efficient learning of geometric structures from single-view images
Promotor: Prof. dr. ir. M. J. T. Reinders
Copromotors: Dr. J. C. van Gemert and Dr. S. L. Pintea

2022-Now Postdoc, Computer Vision Lab
Delft University of Technology, Delft, The Netherlands

List of Publications

- **Yancong Lin**, Silvia-Laura Pintea, and Jan van Gemert. *Deep Hough-Transform Line Priors*. European Conference on Computer Vision (ECCV), 2020. arXiv: https://arxiv.org/abs/2007.09493. (Chapter 2)

- **Yancong Lin**, Silvia-Laura Pintea, and Jan van Gemert. *Semi-Supervised Lane Detection with Deep Hough Transform*. International Conference on Image Processing (ICIP), 2021. arXiv: https://arxiv.org/abs/2106.05094. (Chapter 3)

- **Yancong Lin**, Ruben Wiersma, Silvia-Laura Pintea, Klaus Hildebrandt, Elmar Eisemann, and Jan van Gemert. *Deep vanishing point detection: Geometric priors make dataset variations vanish*. Conference on Computer Vision and Pattern Recognition (CVPR), 2022. arXiv: https://arxiv.org/abs/2203.08586. (Chapter 4)

- **Yancong Lin**, Silvia-Laura Pintea, and Jan van Gemert. *Data-Efficient Learning for 3D Mirror Symmetry Detection*. arXiv: https://arxiv.org/abs/2112.12579. (Chapter 5)

- Andrea Alfieri, **Yancong Lin**, and Jan van Gemert. *Investigating Transformers in the Decomposition of Polygonal Shapes as Point Collections*. International Conference on Computer Vision *Workshops* (ICCVW), 2021. arXiv: https://arxiv.org/abs/2108.07533. (Chapter 6)