Deep Hough-Transform Line Priors

Yancong Lin, Silvia L. Pintea, and Jan C. van Gemert

Computer Vision Lab
Delft University of Technology, the Netherlands

Abstract. 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.
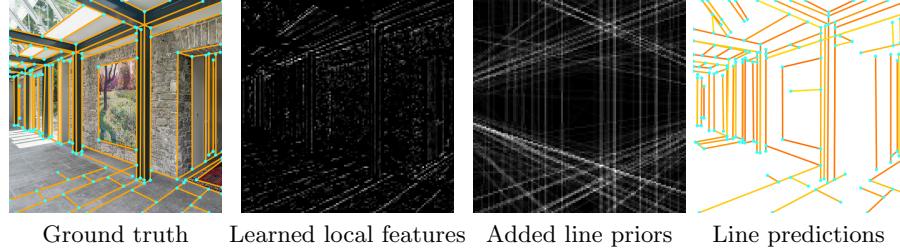
Keywords: Hough transform; global line prior, line segment detection.

1 Introduction

Line segment detection is a classic Computer Vision task, with applications such as road-line detection for autonomous driving [17,22,30,36], wireframe detection for design in architecture [18,54,55], horizon line detection for assisted flying [12,32,39], image vectorization [41,56]. Such problems are currently solved by state-of-the-art line detection methods [18,54,51] by relying on deep learning models powered by huge, annotated, datasets.

Training deep networks demands large datasets [2,35], which are expensive to annotate. The amount of needed training data can be significantly reduced by adding prior knowledge to deep networks [5,19,21]. 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 [21,43], and multi-scale filters add a scale-invariance prior [37,40]. 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 [10,20,26]. It parameterizes lines in terms of two geometric terms: an offset and an angle, describing the line

Ground truth   Learned local features   Added line priors   Line predictions

Fig. 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.

equation in polar coordinates. This gives a global representation for every line in the image. As shown in figure 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 [6,7,31,44] nor do we learn everything in deep architectures [18,51,54]. 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) [18] and York Urban [8].

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 [31,44], connected components [6], probabilistic graphical models [7]. 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 [10]. Given its simplicity and effectiveness, subsequent line-detection work followed this approach [11,20,49], by focusing on analyzing peaks in Hough space. To overcome the sensitivity to noise, previous work proposed statistical analysis of Hough space [50], and segment-set selection based on hypothesis testing [45]. Similarly, a probabilistic Hough transform for line detection, followed by Markov Chain modelling of candidate lines is proposed in [1], while [26] creates a progressive probabilistic Hough transform, which is both faster and
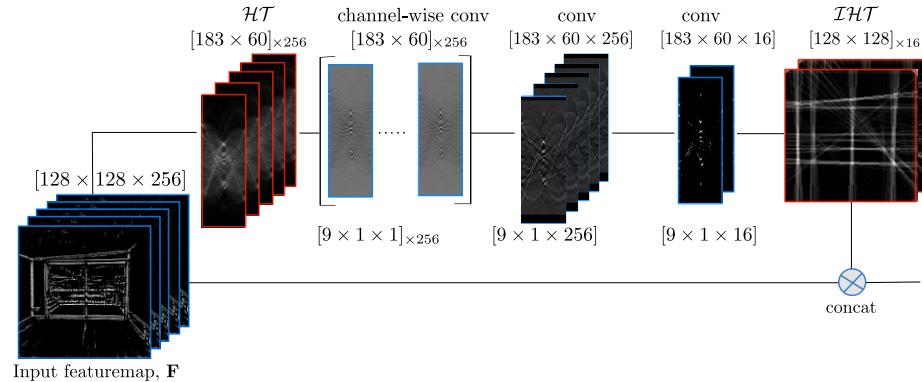
more robust to noise. An extension of Hough transform with edge orientation is used in [13]. Though less common, the slope-intercept parameterization of Hough transform for detecting lines is considered in [38]. In [29] Hough transform is used for detecting page orientation for character recognition. 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 [18] uses two heads: one for junction prediction and one for line detection. This is extended in [54], by a line-proposal sub-network. A segmentation-network backbone combined with an attraction field map, where pixels vote for their closest line is used in [51]. Similarly, attraction field maps are also used in [52] for generating line proposals in a deep architecture. Applications of line prediction using a deep network include aircraft detection [46], and power-line detection [28]. Moving from 2D to 3D, [55] 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 [51] which creates a fisheye-distorted wireframe dataset and proposes a method to rectify it. A graph formulation [53] can learn the association between end-points. The need for geometric priors for horizon line detection is investigated in [48], 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 [4] for curve detection. Deep Hough voting schemes are considered in [33] for detecting object centroids on 3D point clouds, and for finding image correspondences [27]. 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 [3,23,42]. Inverting the Radon transform back to the image domain is considered in [14,34]. In [34] an exact inversion from partial data is used, while [14] relies on a deep network for the inversion, however the backprojection details are missing. Related to Radon transform, the ridgelet transform [9] maps points to lines, and the Funnel transform detects lines by accumulating votes using the slope-intercept line representation [47]. Similar to these works, we take inspiration from the Radon transform and its inversion in defining our Hough transform block.

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 3.(a,b) and [10].

Fig. 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.
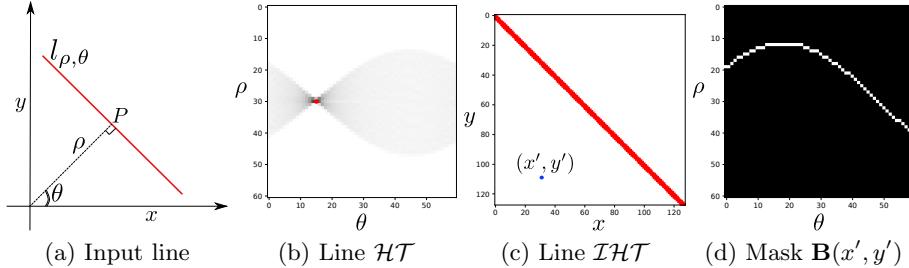
We present a Hough transform and inverse Hough transform (HT-IHT block) to combine local learned image features with global line priors. We allow the network to combine information by defining the Hough transform on a separate residual branch. The $\mathcal{HT}$ layer inside the HT-IHT block maps input 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 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.

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 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 by:

$$(x(i), y(i)) = (\rho \cos\theta - i \sin\theta, \rho \sin\theta + i \cos\theta), \tag{1}$$

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.

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

Fig. 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.

The traditional Hough transform [10,26] uses binary input where featuremaps are real valued. Instead of binarizing the featuremaps, we define the Hough transform similar to the Radon transform [3]. 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:

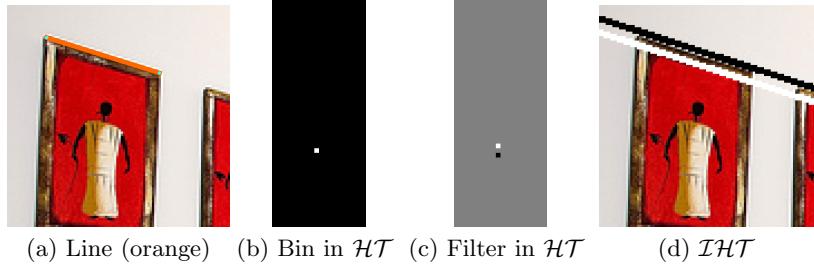$$\mathcal{HT}(\rho, \theta) = \sum_i \mathbf{F}_{\rho,\theta}(x(i), y(i)), \tag{2}$$

where the relation between the pixel $(x(i), y(i))$ and bin (ρ, θ) is given in equation (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 3.(b) shows the Hough transform map for the input line in figure 3.(a), where we highlight in red the bin corresponding to the line.

Note that in equation (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:

$$\mathcal{HT} = \mathbf{FB}. \tag{3}$$

For gradient stability, we additionally 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) $\mathcal{IHT}$

Fig. 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.

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 pixel locations in the input featuremap, $\mathbf{F}$. Following [3], 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:

$$\mathcal{IHT}(x, y) = \frac{1}{N_\theta} \sum_\theta \mathcal{HT}(x \cos \theta + y \sin \theta, \theta). \tag{4}$$

In the backward pass, $\frac{\partial \mathcal{HT}}{\partial F(x,y)}$, we use equation (4) without the normalization over the number of angles, N_θ.
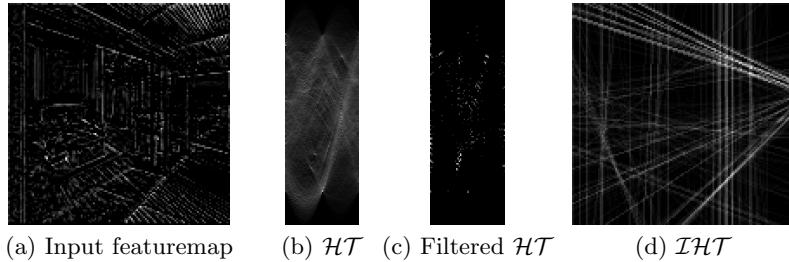
Similar to the forward Hough transform pass, we store the correspondence between the pixels in the input featuremap (x, y) and the Hough transform bins (ρ, θ), in the binary matrix, $\mathbf{B}$. We implement the inverse Hough transform as a matrix multiplication of $\mathbf{B}$ with the learned $\mathcal{HT}$ map, for each channel:

$$\mathcal{IHT} = \mathbf{B} \left(\frac{1}{N_\theta} \mathcal{HT} \right). \tag{5}$$

Figure 3.(c) shows the $\mathcal{IHT}$ of the Hough transform map in figure 3.(b), while figure 3.(d) shows the binary mask $\mathbf{B}$ for the pixel (x', y') highlighted in blue in figure 3.(c), mapping it to its corresponding set of bins in the Hough map.

3.3 Convolution in Hough Transform space

Local operations in Hough space correspond to global operations in the image space, see figure 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 [23]. The filtering in the Hough domain is done locally over the offsets, for each angle direction [29,46]. We perform channel-wise $1D$ convolutions in the Hough space

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

Fig. 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.

over the offsets, ρ, as the Hough transform is also computed channel-wise over the input featuremaps. In Figure 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.

Inspired by the Radon literature [23,29,46] 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:

$$f(\rho) \stackrel{init}{=} -\frac{\partial^2 g(\rho, \sigma)}{\partial \rho^2}, \tag{6}$$

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.
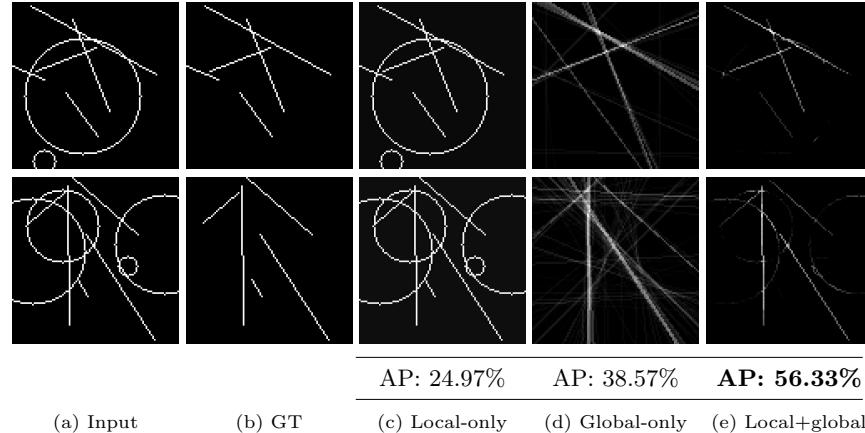
4   Experiments

We conduct experiments on three datasets: a controlled Line-Circle dataset, the Wireframe (ShanghaiTech) [18] dataset and the York Urban [8] 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].

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

---

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

| | | AP: 24.97% | AP: 38.57% | **AP: 56.33%** |
| (a) Input | (b) GT | (c) Local-only | (d) Global-only | (e) Local+global |

Fig. 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.

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 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 [18,25,24] 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 [25].

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 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.

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 1. Exp 2: The effect of convolution in the Hough domain, in terms of AP on a subset of the Wireframe (ShanghaiTech) dataset [18]. 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.
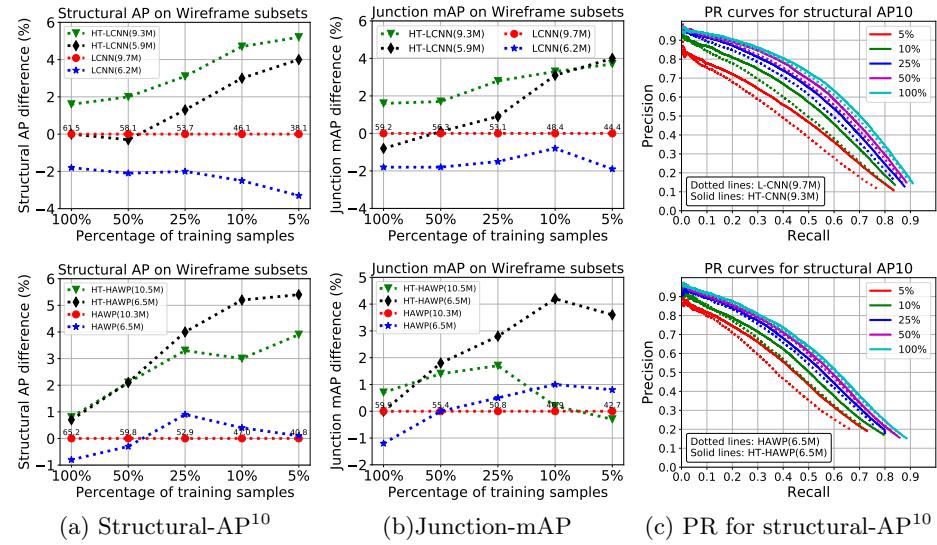
4.2   Exp 2: The effect of convolution in the Hough domain

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 [18]. 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 [55], 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 [16] 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 binary images with line segments. We insert our HT-IHT block after every residual block. All layers are initialized with the He initialization [15].

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 for merging and reducing the channels, and (4) using three standard 3×3 convolutions.

Experimental analysis. Table 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 [15]. 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.

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

Fig. 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 [54] and HAWPs [52]. 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.

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) [18] and York Urban [8] 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 [55]. We also report precision-recall, following [1], 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 [1]. 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 [54] and HAWP [52] 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 [52,55]: optimizing binary cross-entropy loss for junction and line prediction, and L_1 loss for junction offsets. The training uses the ADAM
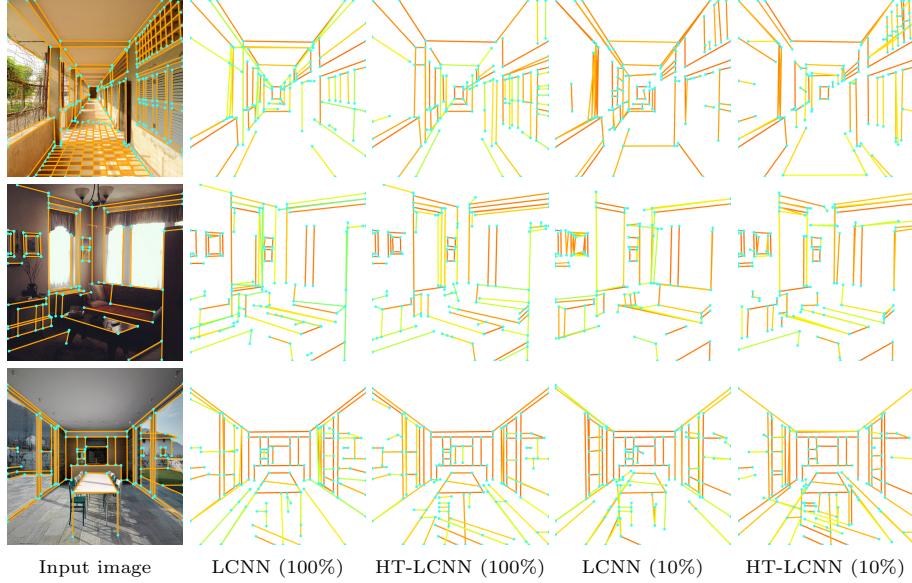
optimizer, with scheduled learning rate starting at $4e - 4$, and $1e - 4$ weight decay, for a maximum of 30 epoch.

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 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 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 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.
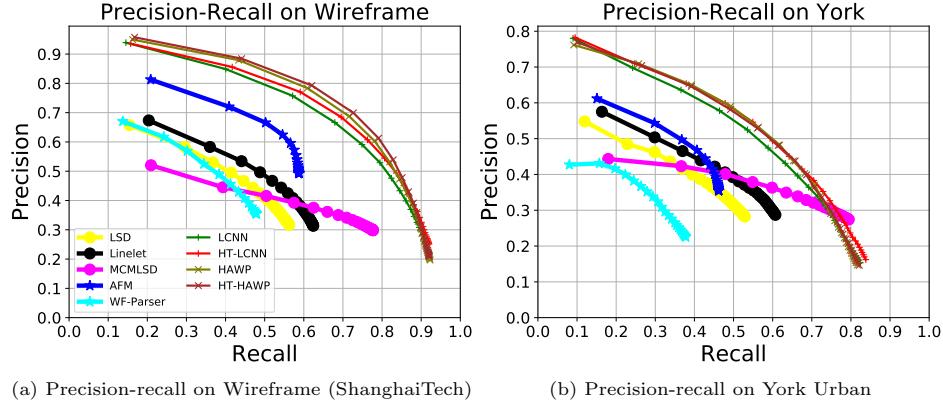
Exp 3.(b): Comparison with state-of-the-art. We compare our HT-LCNN and HT-HAWP, starting from LCNN [54] and HAWP [52] and using HT-IHT blocks instead of the hourglass blocks, with five state-of-the-art models on the Wireframe (ShanghaiTech) [18] and York Urban [8] datasets. The official training split 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 [44], Linelet [7] and MCMLSD [1], and four learning-based methods: AFM [51], WF-Parser (Wireframe Parser) [18], LCNN [54], HAWP [52]. 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.

| Input image | LCNN (100%) | HT-LCNN (100%) | LCNN (10%) | HT-LCNN (10%) |

Fig. 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 supplementary material for more results).

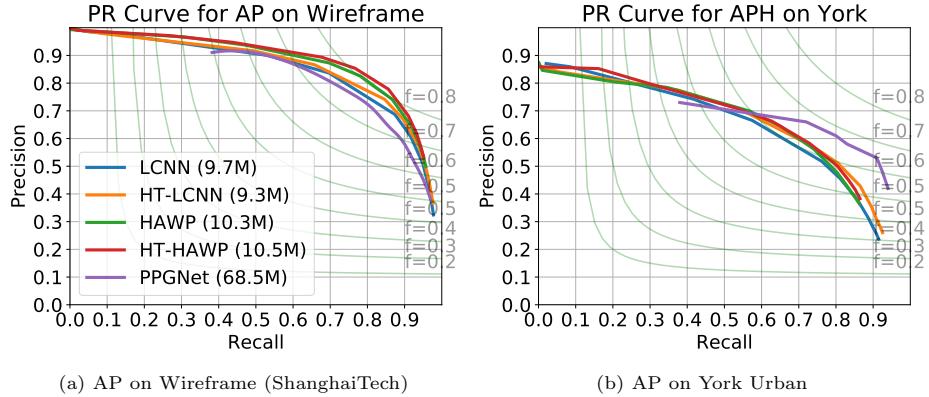| Train/test			Wireframe / Wireframe			Wireframe / York Urban		
			Structural		Junction	Structural		Junction
Metrics	#Params	FPS	AP^5	AP^{10}	mAP	AP^5	AP^{10}	mAP
LSD [44]	—	15.3	7.1	9.3	16.5	7.5	9.2	14.9
Linelet [7]	—	0.04	8.3	10.9	17.4	9.0	10.8	18.2
MCMLSD [1]	—	0.2	7.6	10.4	13.8	7.2	9.2	14.8
WF-Parser [18]	31 M	1.7	6.9	9.0	36.1	2.8	3.9	22.5
AFM [51]	43 M	6.5	18.3	23.9	23.3	7.1	9.1	12.3
LCNN [54]	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 [52]	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. 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.

(a) Precision-recall on Wireframe (ShanghaiTech)   (b) Precision-recall on York Urban

Fig. 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.

Table 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 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 9 shows precision-recall scores [1] on the Wireframe (ShanghaiTech) and York Urban datasets. MCMLSD [1] shows good performance in the high-recall zone on the York Urban dataset, but its performance is lacking in the low-recall zone. AFM [51] 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

(a) AP on Wireframe (ShanghaiTech)

(b) AP on York Urban

Fig. 10. Exp 3.(b): Comparing PPGNet[53] 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.

competitive performance when compared to state-of-the-art models, thus validating the usefulness of the HT-IHT block.

In figure 10, we compare our HT-LCNN and HT-HAWP with PPGNet [53]. 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 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.

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

References

1. Almazan, E.J., Tal, R., Qian, Y., Elder, J.H.: Mcmlsd: A dynamic programming approach to line segment detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2031–2039 (2017)
2. Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., Katz, B.: Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In: Advances in Neural Information Processing Systems. pp. 9448–9458 (2019)
3. Beatty, J.: The Radon Transform and the Mathematics of Medical Imaging. Honors thesis, Digital Commons @ Colby (2012)
4. Beltrametti, M.C., Campi, C., Massone, A.M., Torrente, M.L.: Geometry of the hough transforms with applications to synthetic data. CoRR (2019)
5. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1872–1886 (2013)
6. Burns, J.B., Hanson, A.R., Riseman, E.M.: Extracting straight lines. IEEE transactions on pattern analysis and machine intelligence (4), 425–455 (1986)
7. Cho, N.G., Yuille, A., Lee, S.W.: A novel linelet-based representation for line segment detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(5), 1195–1208 (2017)
8. Denis, P., Elder, J.H., Estrada, F.J.: Efficient edge-based methods for estimating manhattan frames in urban imagery. In: European Conference on Computer Vision. pp. 197–210. Springer (2008)
9. Do, M.N., Vetterli, M.: The finite ridgelet transform for image representation. IEEE Transactions on image Processing **12**(1), 16–28 (2003)
10. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Communications of the ACM **15**(1), 11–15 (1972)
11. Furukawa, Y., Shinagawa, Y.: Accurate and robust line segment extraction by analyzing distribution around peaks in hough space. Computer Vision and Image Understanding **92**(1), 1–25 (2003)
12. Gershikov, E., Libe, T., Kosolapov, S.: Horizon line detection in marine images: which method to choose? International Journal on Advances in Intelligent Systems **6**(1) (2013)
13. Guerreiro, R.F., Aguiar, P.M.: Connectivity-enforcing hough transform for the robust extraction of line segments. IEEE Transactions on Image Processing **21**(12), 4819–4829 (2012)
14. He, J., Ma, J.: Radon inversion via deep learning. In: Medical Imaging (2018)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Hillel, A.B., Lerner, R., Levi, D., Raz, G.: Recent progress in road and lane detection: a survey. Machine vision and applications **25**(3), 727–745 (2014)
18. Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 626–635 (2018)
19. Jacobsen, J.H., van Gemert, J., Lou, Z., Smeulders, A.W.: Structured receptive fields in cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2610–2619 (2016)

20. Kamat-Sadekar, V., Ganesan, S.: Complete description of multiple line segments using the hough transform. Image and Vision Computing **16**(9-10), 597–613 (1998)
21. Kayhan, O.S., van Gemert, J.C.: 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. pp. 14274–14285 (2020)
22. Lee, S., Kim, J., Shin Yoon, J., Shin, S., Bailo, O., Kim, N., Lee, T.H., Seok Hong, H., Han, S.H., So Kweon, I.: Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In: Proceedings of the IEEE international conference on computer vision. pp. 1947–1955 (2017)
23. Magnusson, M.: Linogram and Other Direct Fourier Methods for Tomographic Reconstruction. Linköping studies in science and technology: Dissertations, Department of Mechanical Engineering, Linköping University (1993)
24. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
25. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE transactions on pattern analysis and machine intelligence **26**(5), 530–549 (2004)
26. Matas, J., Galambos, C., Kittler, J.: Robust detection of lines using the progressive probabilistic hough transform. Computer Vision and Image Understanding **78**(1), 119–137 (2000)
27. Min, J., Lee, J., Ponce, J., Cho, M.: Hyperpixel flow: Semantic correspondence with multi-layer neural features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3395–3404 (2019)
28. Nguyen, V.N., Jenssen, R., Roverso, D.: Ls-net: Fast single-shot line-segment detector. CoRR (2019)
29. Nikolaev, D.P., Karpenko, S.M., Nikolaev, I.P., Nikolayev, P.P.: Hough transform: underestimated tool in the computer vision field. In: Proceedings of the 22th European Conference on Modelling and Simulation. vol. 238, p. 246 (2008)
30. Niu, J., Lu, J., Xu, M., Lv, P., Zhao, X.: Robust lane detection using two-stage feature extraction with curve fitting. Pattern Recognition **59**, 225–233 (2016)
31. Pătrăucean, V., Gurdjos, P., Von Gioi, R.G.: A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In: European Conference on Computer Vision. pp. 572–585 (2012)
32. Porzi, L., Rota Bulò, S., Ricci, E.: A deeply-supervised deconvolutional network for horizon line detection. In: Proceedings of the 24th ACM international conference on Multimedia. pp. 137–141 (2016)
33. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9277–9286 (2019)
34. Rim, D.: Exact and fast inversion of the approximate discrete radon transform from partial data. Applied Mathematics Letters **102**, 106159 (2020)
35. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
36. Satzoda, R.K., Trivedi, M.M.: Efficient lane and vehicle detection with integrated synergies (elvis). In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 708–713 (2014)
37. Shelhamer, E., Wang, D., Darrell, T.: Blurring the line between structure and learning to optimize and adapt receptive fields. CoRR (2019)

38. Sheshkus, A., Ingacheva, A., Arlazarov, V., Nikolaev, D.: Houghnet: neural network architecture for vanishing points detection. International Conference on Document Analysis and Recognition (ICDAR) (2019)
39. Simon, G., Fond, A., Berger, M.O.: A-contrario horizon-first vanishing point detection using second-order grouping laws. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 318–333 (2018)
40. Sosnovik, I., Szmaja, M., Smeulders, A.: Scale-equivariant steerable networks. International Conference on Learning Representations (2020)
41. Sun, J., Liang, L., Wen, F., Shum, H.Y.: Image vectorization using optimized gradient meshes. ACM Transactions on Graphics (TOG) **26**(3), 11–es (2007)
42. Toft, P.: The Radon Transform: Theory and Implementation. Department of Mathematical Modelling, Section for Digital Signal Processing, Technical University of Denmark (1996)
43. Urban, G., Geras, K.J., Kahou, S.E., Aslan, O., Wang, S., Caruana, R., Mohamed, A., Philipose, M., Richardson, M.: Do deep convolutional nets really need to be deep and convolutional? International Conference on Learning Representations (2016)
44. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2008)
45. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: On straight line segment detection. Journal of Mathematical Imaging and Vision **32**(3), 313 (2008)
46. Wei, H., Bing, W., Yue, Z.: X-linenet: Detecting aircraft in remote sensing images by a pair of intersecting line segments. CoRR (2019)
47. Wei, Q., Feng, D., Zheng, W.: Funnel transform for straight line detection. CoRR (2019)
48. Workman, S., Zhai, M., Jacobs, N.: Horizon lines in the wild. British Machine Vision Conference (2016)
49. Xu, Z., Shin, B.S., Klette, R.: Accurate and robust line segment extraction using minimum entropy with hough transform. IEEE Transactions on Image Processing **24**(3), 813–822 (2014)
50. Xu, Z., Shin, B.S., Klette, R.: A statistical method for line segment detection. Computer Vision and Image Understanding **138**, 61–73 (2015)
51. Xue, N., Bai, S., Wang, F., Xia, G.S., Wu, T., Zhang, L.: Learning attraction field representation for robust line segment detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (June 2019)
52. Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.S., Zhang, L., Torr, P.H.: Holistically-attracted wireframe parsing. In: Conference on Computer Vision and Pattern Recognition (2020)
53. Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., Gao, S.: Ppgnet: Learning point-pair graph for line segment detection. In: Conference on Computer Vision and Pattern Recognition (2019)
54. Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 962–971 (2019)
55. Zhou, Y., Qi, H., Zhai, Y., Sun, Q., Chen, Z., Wei, L.Y., Ma, Y.: Learning to reconstruct 3d manhattan wireframes from a single image. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7698–7707 (2019)
56. Zou, J.J., Yan, H.: Cartoon image vectorization based on shape subdivision. In: Proceedings. Computer Graphics International 2001. pp. 225–231 (2001)