Received February 23, 2022, accepted April 4, 2022, date of publication April 21, 2022, date of current version May 3, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169177

LSDNet: Trainable Modification of LSD Algorithm for Real-Time Line Segment Detection

LEV TEPLYAKOV[1], LEONID ERLYGIN[1,2], AND EVGENY SHVETS[1]
[1]Institute for Information Transmission Problems, Russian Academy of Sciences, 119991 Moscow, Russia
[2]Department of Control and Applied Mathematics, Moscow Institute of Physics and Technology, 117303 Moscow, Russia

Corresponding author: Lev Teplyakov (teplyakov@iitp.ru)

This work was supported by the Russian Science Foundation Project No. 20-61-47089.

ABSTRACT As of today, the best accuracy in line segment detection (LSD) is achieved by algorithms based on convolutional neural networks – CNNs. Unfortunately, these methods utilize deep, heavy networks and are slower than traditional model-based detectors. In this paper we build an accurate yet fast CNN-based detector, LSDNet, by incorporating a lightweight CNN into a classical LSD detector. Specifically, we replace the first step of the original LSD algorithm – construction of line segments heatmap and tangent field from raw image gradients – with a lightweight CNN, which is able to calculate more complex and rich features. The second part of the LSD algorithm is used with only minor modifications. Compared with several modern line segment detectors on standard Wireframe dataset, the proposed LSDNet provides the highest speed (among CNN-based detectors) of 214 FPS with a competitive accuracy of 78 F^H. Although the best-reported accuracy is 83 F^H at 33 FPS, we speculate that the observed accuracy gap is caused by errors in annotations and the actual gap is significantly lower. We point out systematic inconsistencies in the annotations of popular line detection benchmarks – Wireframe and York Urban, carefully reannotate a subset of images and show that (i) existing detectors have improved quality on updated annotations without retraining, suggesting that new annotations correlate better with the notion of correct line segment detection; (ii) the gap between accuracies of our detector and others diminishes to negligible 0.2 F^H, with our method being the fastest.

INDEX TERMS Convolutional neural networks, edge detection, line segment detection, U-net, LSD.

I. INTRODUCTION

Automatic general-purpose line segment detection is a long-standing computer vision problem of high practical importance. Line segment detectors are exploited to construct an intermediate representation of image contents in visual recognition systems over a wide range of applications, such as autonomous vehicle localization [1]–[3], infrastructure maintenance with an UAV [4], [5], document recognition [6], [7].

Traditionally, the problem of line segment detection was approached with so-called model-based algorithms [8]–[11]. These algorithms operate by searching an image for elements that satisfy an explicit definition of a salient line segment, for example, "line segment is a strip-like set of image pixels with similar gradients" [8], [9] or "an image region is a line segment if its contour map triggers a peak in Hough space" [10], [11]. These algorithms typically have the benefits of being fast and having interpretable parameters. However, they may miss the segments that are salient for human, but for some reason don't match the exact explicit implemented definition. They are also prone to over-segmentation (splitting a single segment into parts) and sometimes demand nontrivial problem-specific postprocessing [12], [13].

The troubling problem of formulating an explicit criterion that matches the human expectation of what exactly constitutes a "salient line segment" can be avoided by manually annotating images and training a CNN, which then learns an implicit algorithm from data samples. This is an approach that yields the best accuracy in line segment detection task today [14]–[20]. Skipping ahead, let us note that such annotation is not a simple task either - existing datasets on line segment detection have numerous and sometimes extreme internal inconsistencies - probably caused by the inherent ambiguity of the task, lack of clear labeling instructions and the tediousness of the task, leading to missed segments.

The associate editor coordinating the review of this manuscript and approving it for publication was Sudhakar Radhakrishnan.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

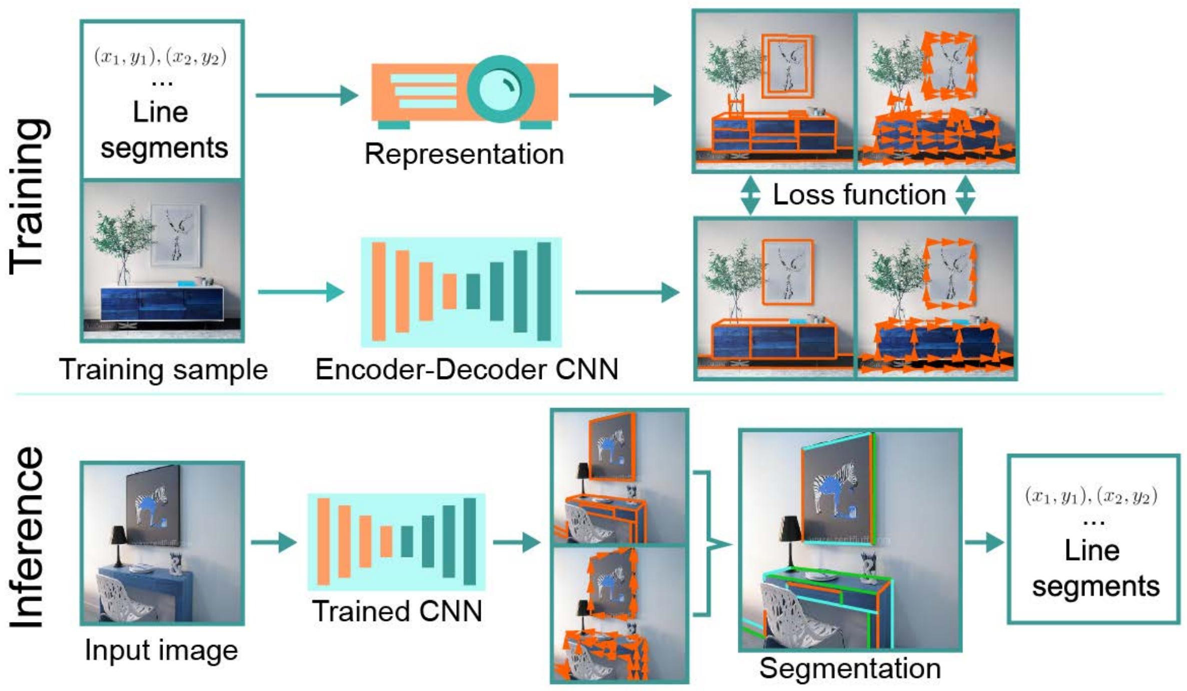

FIGURE 1. Overview of the proposed approach. A lightweight neural network predicts line segment mask and tangent vector field, which are then clustered and each cluster is represented as a line segment.

From a technical perspective, there also is a challenge in designing a CNN-based line detector. The typical solution is that CNN constructs an intermediate representation - encoding - which is then converted into a set of answers by some hand-crafted algorithm. Object detection networks have solved this problem by using so-called anchors [21] and, later, more simple anchorless detectors such as FCOS [22]. But line segment detectors need alternative encodings, suitable not for bounding boxes, but for line segments. The encodings should effectively deal with the fact that the line segments to be detected in a typical image intersect with each other a lot, while encodings for bounding boxes are effective only when "overlapping mostly happens between objects with considerably different sizes" [22], making them hard to exploit for line segment detection. While object detectors encondings - after many iterations of refinement - have become fast and elegant, we believe that intermediate representations of most line detector used today are still either imprecise, slow or unintuitive.

So whether it is the complexity of the interpreter or the sheer weight of the CNN backbone, CNN-based detectors that outperform the traditional ones in accuracy are also computationally harder [20]. Their complexity limits the scope of application of such algorithms in cases where speed, energy consumption, or hardware price are critical.

In this work we propose a fast yet accurate CNN-based line segment detection algorithm, LSDNet, built on the basis of a widely used model-based detector, LSD [8]. LSDNet overview is presented in Figure 1. The first step of LSD is the calculation of image gradient's orientation and magnitude. We view this step as an estimation of an intermediate representation, composed of line segments' heatmap, estimated as gradient's magnitude, and tangent field, estimated as gradient's orientation. We substitute this step with a lightweight CNN to generate the heatmap and tangent field from a more diverse and complex set of features than a simple gradient; the second step, conversion the intermediate representation into a set of line segments, is taken from the LSD almost as-is. This substitution boosts the LSD accuracy and simplifies the postprocessing due to more accurate heatmap and tangent field. The heatmap and tangent field generation is a relatively simple task - the answer can be correctly inferred from the local context - which allows to use a lightweight CNN.

We compare LSDNet and a selection of existing detectors: LSD [8], L-CNN [18], HAWP [15], TP-LSD [17], M-LSD and M-LSD-tiny [20] and show that LSDNet provides competitive accuracy of 78 F^H score on Wireframe dataset while being the fastest: 214 FPS for 288×288 input on conventional hardware. LSDNet outperforms the second fastest approach, M-LSD-tiny [20], both in accuracy and speed.

We discuss considerable inconsistencies (section IV) in ground truth labeling of current benchmarks for line segment detection accuracy - datasets Wireframe [23] and YorkUrban [24]. The labeling in these datasets is inconsistent not only between images: within the very same image many segments, similar in appearance, are often marked up differently - some as a positives, others as negatives. We speculate that these datasets in their current state are flawed for assessing the accuracy of general purpose line segment detectors.

So, in addition to measuring the accuracy of our detector on these datasets, we select and reannotate a part of Wireframe dataset - consisting mainly of simple images with less ambiguity - and show that the reannotated subset correlates better with conventional notion of correct line segment detection. Specifically, it narrows the accuracy gap between LSD and CNN-based approaches, and makes the gap between the proposed LSDNet and L-CNN [18], one the most accurate CNN-based approaches, negligible.

II. RELATED WORK

In this section we cover the model-based LSD algorithm [8] and the existing CNN-based approaches to the problem of line segment detection.

A. LSD DETECTOR

LSD is one of the most popular general-purpose line segment detectors and serves as a common baseline for CNN-based algorithms both in accuracy and speed [15]–[17].

The first step of LSD detector is gradient calculation; then the algorithm builds so-called line support regions (LSRs) [25]. LSRs are image segments (not to be confused with line segments), spanning actual line segments in an image. They are built by iteratively grouping neighbouring pixels with high gradients' magnitudes and similar orientations. After the formation of initial LSRs, they undergo several steps of filtering and refinement. These include, among others, splitting LSRs of ''hockey stick'' shape - effectively decoupling distinct but merged regions; removal of LSRs with a high deviation of gradients' orientations - possibly false-positives. Typical resulting LSRs are long, straight and a few pixels thick. Finally, each such region is individually encoded as a pair of points - effectively, a line segment.

LSD detector is fast, reaching 185 FPS for 320×320 images on a conventional CPU, and provides an accuracy of 63.3 F^H on the Wireframe dataset.

B. CNN-BASED APPROACHES

The CNN-based approaches are typically composed of two modules: the CNN itself predicts an intermediate representation, then a postprocessing module reconstructs line segments from this representation.

We consider the design of the intermediate representation to be the key growth point of CNN-based line segment detectors since the desired detector's output - a unknown-size set of line segments with potentially high overlap - is hard to represent as a ''CNN-friendly'' fixed-shape tensor [26]. The survey below covers most popular intermediate representations used in existing CNN-based approaches.

The first CNN-based detectors represented line segments in an image as a set of endpoints and their connectivity graph [18], [23]. The endpoints were detected as local maxima of a CNN-produced heatmap. The connectivity of the endpoints was deduced either with the help of edge map heuristics [23] or with a trainable classifier [18]. To generate the classifier's input, a fixed number of uniform spaced points was sampled from the feature map between the endpoints. This operation was called LoI pooling [18].

In [14] a representation called attraction field was proposed (distance field [19] is a similar concept). Line segments were represented as a 2D vector field of translations to a nearest line segment. The postprocessing step for such a representation required nontrivial line segments extraction from heatmap-like prediction. Interestingly, the postprocessing of even a perfect attraction field generated from dataset annotations did not provide absolute detection accuracy [14] - in other words, such representation is inherently ambiguous.

The covered representations of endpoints and attraction field were combined and modified in [15]. The proposed CNN predicted both the endpoints and the attraction field, which was enriched to encode the translations to the both segment's endpoints as 4D vector field. Then the endpoints and the attraction field were used to refine each other. The refined line segments proposals underwent final verification with the help of LoI pooling and a trainable classifier. The detector provides state-of-the-art quality of 83 F^H on Wireframe [23] dataset to date, but with low speed of 33 FPS on GPU.

The recently proposed ''tri-points'' representation [17] is focused on speeding up the detector. A line segment was represented by its center point and two vectors to its endpoints. It allowed to significantly boost the speed up to 50 FPS, driven by a much faster postprocessing requiring trivial conversion from ''tri-point'' to a line segment and non-maximum suppression. The CNN itself remained comparably slow. In [20] some further enhancements were proposed. A lightweight CNN was designed and the training procedure was improved by augmentations and more sophisticated loss function. It resulted in the fastest CNN-based detector to date with 200 FPS overall and 241 FPS for standalone CNN.

III. PROPOSED APPROACH

A perfect line segments representation should make it possible to design a CNN and a postprocessing module both being fast and accurate. We argue that in the search for such a representation there is no need to develop a brand new one from scratch; instead, the representation used implicitly by LSD detector - line segments heatmap and tangent field - already possesses all the desired properties. Indeed, the heatmap and tangent field could be inferred from local image context and does not require the reasoning of complex abstract features, which allows to use a lightweight CNN. On the other hand, as proven by LSD, the representation could be efficiently postprocessed to actual line segments. In the next sections we cover each step of the algorithm in detail.

A. CNN

1) LINE SEGMENT REPRESENTATION

We represent a set of line segments in an image as a 2-channel feature map of the same height and width as the image. The first channel denoted by M contains a line segment mask.

The second channel denoted as F contains tangent vector field of line segments. Since the values of F are unit vectors, we encode F as one channel feature map.

Let $l = (x_1, y_1, x_2, y_2)$ be a line segment, φ_l - segment's level line angle in range $[0, \pi)$, $p = (x, y)$ - an image pixel, $L_p = l_1, l_2, \ldots$ - set of line segments crossing p. Then if $L_p = \emptyset$, then $M(p) = 0$ and $F(p)$ is arbitrary; otherwise $M(p) = 1, F(p) = \varphi_{l_1}$.

Note that in case of overlapping segments ($||L_p|| > 1$) $F(p)$ is defined by one arbitrary segment of overlap. This ambiguity probably can be ignored since only about 1% of line segments' pixels lie on overlaps - as measured on Wireframe dataset [23].

2) LOSS FUNCTION

The loss function $L = L_{mask} + \alpha L_{field}$ used to fit the network is composed of two independent weighted terms, one responsible for mask M, other - for the tangent vector field F.

While the prediction of mask M is a straightforward segmentation problem with L_{mask} being a conventional cross entropy loss, to correctly estimate error in prediction of the tangent field F we should account for the following property of line segments' level line angles - the distance between angles 0^0, 10^0 and 0^0, 170^0 should be equal.

This problem can be approached by computing several distances between the original angles and the angles, shifted by $\pm\pi$, and picking the minimum distance [27]. The calculation can be done simpler: let φ_1, φ_2 - angles between which the distance is to be computed, $z_1 = e^{i\varphi_1}, z_2 = e^{i\varphi_2} \in \mathbf{C}$ - the representation of the angles as complex numbers with unit length and phases φ_1, φ_2, then

$$\rho(\varphi_1, \varphi_2) = ||z_1^2 - z_2^2||_2 \qquad (1)$$

This distance function has a geometrical interpretation, illustrated in Fig. 2a. It equals to ℓ^2-norm of vector difference between unit vectors with phases $2\varphi_1$, $2\varphi_2$. Turning to the aforementioned example, the doubled phase makes vectors, corresponding to angles 10^0 and 170^0, equally close to the horizontal, corresponding to angle 0^0.

Given the angle distance function ρ, the predicted field F_p, the reference mask M_t and reference field F_t, the tangent vector field loss L_{field} is defined as

$$L_{field} = \frac{1}{\sum_p M_t(p)} \sum_{p:M_t(p)=1} \rho^2(F_t(p), F_p(p)), \qquad (2)$$

where M_t is the reference line segment mask - essentially, loss is the average tangent angle discrepancy over the pixels that correspond to the ground truth line segments.

3) CNN ARCHITECTURE

To predict the proposed feature map, we use a CNN of U-Net [28] family. The architecture we use differs from the original one in the following simplifications. We exploit padded convolutions providing the same input and output spatial sizes of convolutional layers, which allows not to crop the feature maps feeded to skip connections. Instead of transposed convolutions we use bilinear upsampling. We reduce the depth of encoder-decoder branches up to 3 maxpooling and 3 upsampling layers, correspondingly, and use fewer filters in convolutional blocks - 16, 32, 64, 128 filters per block (the number of blocks is greater than the number if maxpooling layers by one). The resulting CNN has $\approx$ 0.5M trainable parameters and can run at 48 FPS on CPU and at 695 FPS on GPU (refer to section V for benchmarking details).

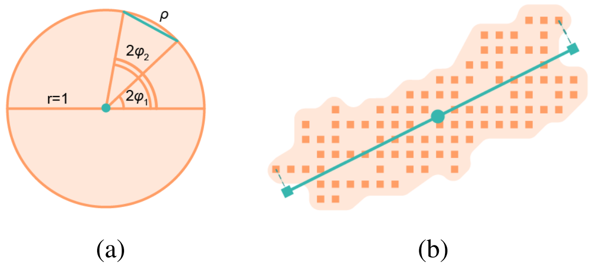

FIGURE 2. (a) A visualization of the distance function between angles φ_1, φ_2. It accounts for the periodicity of angles. (b) Line segment extraction from a region.

B. LINE SEGMENTS RECONSTRUCTION

Let us consider how the predicted segments mask M and tangent field F are converted into the desired output - a set of line segment $((x_1, y_1), (x_2, y_2)), \ldots$. This process has three steps: firstly, the predicted features are coarsely segmented into lines and background (section "Foreground segmentation"). Secondly, the lines are finely segmented into several line support regions (LSRs) (section "Region grouping"). Finally, a line segment and its confidence is extracted from each LSR (section "Line segments extraction").

1) FOREGROUND SEGMENTATION

The first step of line segments reconstruction is to segment foreground (lines) from background (not lines).

We use a coarse-to-fine binarization approach by multiplying the masks of global thresholding $M(p) > \tau$ and local thresholding

$$M(p) > \sum_{d \in K(p)} W_p(d) M(d) - \theta \qquad (3)$$

where θ - threshold, K_p - a window centered at pixel p, $W_p(d)$ - Gaussian averaging weight.

Global thresholding with a small threshold gives a coarse extraction of line segments mask, but often incorrectly joins close - but separate - line segments. On the contrary, local thresholding provides much finer local distinction of line segments, but can produce clumps of false positive detections in low intensity areas (Fig. 3). The combination of these binarizations by simple multiplication of the resulting masks allows to filter out false positives of both types and achieve better accuracy.

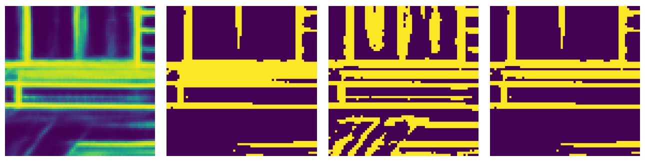

FIGURE 3. Binarization of CNN prediction. Global binarization joins close regions, while local binarization is noisy for the regions of near-zero intensity. The combined binarization is free of both drawbacks.

2) REGION GROUPING

The goal of this step (being a modification of a similar step of LSD algorithm [8]) is to split the foreground, binarized at the previous step, into narrow strip-like LSRs, one per true line segment.

Informally, we want neighbouring pixels p_1, p_2 to be assigned to one LSR, if the values of $M(p_1)$, $F(p_1)$ and $M(p_2)$, $F(p_2)$ are similar. The algorithm grows LSRs iteratively, starting from pixels with highest M value and adding new pixels to the existing LSR which are geometrically close to the pixels and have similar features.

Let us formally introduce the similarity measure used to decide whether pixel g is fit to be joined into a LSR. Let $R = p_1, p_2, \ldots, p_n$ be the set of pixels of this LSR, $I_R = 1/n \cdot \sum_{p \in R} M(p)$ be the mean line segments' mask over it, and $\phi_R = \angle\sqrt{\sum_{p\in R} e^{2iF(p)}}$ - the average tangent field (here $\angle z = \operatorname{atan2}(\operatorname{Im}(z)/\operatorname{Re}(z))$ is the phase of a complex number). Then the similarity function is given by

$$d_{M,F}(g, R) = \rho^2(F(g), \phi_R) + \alpha(M(g) - I_R)^2 \tag{4}$$

The first term defines similarity of tangent field orientation (refer to Eq. (1) for details), the second - the similarity of line mask, α - weighting coefficient. Given the distance function, LSRs are built with an iterative growing algorithm, presented in Algorithm 1.

3) LINE SEGMENTS EXTRACTION

Each LSR $R = p_1, p_2, \ldots, p_n$, should be converted into a line segment satisfying the following criteria.

- The segment goes through LSR's center of mass p_μ

$$p_\mu = \frac{1}{\sum_{p\in R} M(p)} \sum_{p\in R} M(p)p. \tag{5}$$

- The segment is collinear with minor eigenvector a of region's inertia tensor I defines as follows

$$I = \sum_{p\in R} M(p) I^*(p - p_\mu) \tag{6}$$

$$I^*(x, y) = \begin{pmatrix} y^2 & -xy \\ -xy & x^2 \end{pmatrix} \tag{7}$$

- The segment spans the furthest LSR's points, projected onto axis a. The segment's confidence is mean value of M over the region R.

Line segment extraction is visualized in Figure 2b.

Algorithm 1: Region Grouping Algorithm. $d_{M,F}(p, R)$ - Distance Function Between Pixel p and Region R, $N_8(R)$ - 8-Neighbours of Region R

```
input : Line heatmap $M \in [0, 1]^{hw}$, tangent field
        $F \in [0, \pi)^{hw}$
output: Set of line support regions $\mathcal{R} = \{R\}$
param: $\tau \in \mathbf{R}$ - distance threshold
param: $n \in \mathbf{N}$ - minimum region size
$B \in \{0, 1\}^{hw}$ - foreground segmentation of $M$;
$P = (p_i)$, such that $B(p) = 1, M(p_i) \geq M(p_{i+1})$;
$P_{used} \leftarrow \varnothing$;
for $p \in P$ do
    if $p \in P_{used}$ then
        continue;
    end
    $R \leftarrow p$;
    do
        $region\_updated \leftarrow False$;
        for $g \in N_8(R) \setminus P_{used}$ do
            if $d_{M,F}(g, R) < \tau$ then
                Add $R \leftarrow g$;
                Add $P_{used} \leftarrow g$;
                $region\_updated \leftarrow True$;
            end
        end
    while $region\_updated$;
    Add $\mathcal{R} \leftarrow R$;
end
for $R \in \mathcal{R}$ do
    if $|R| < n$ then
        remove $R$ from $\mathcal{R}$;
    end
end
```

IV. DATASET

In this section we analyze the issues of the existing line segment detection datasets and propose a dataset Wireframe-tiny++, a subset of Wireframe dataset [23] with refined annotations.

A. THE EXISTING DATASETS

To the best of our knowledge, there are two widely-used public line segment detection datasets: Wireframe [23] and YorkUrban [24]. The former is composed of 5.000 train and 462 test images, the latter is composed of 120 test images. The datasets contain both indoor and outdoor colour images of various man-made environments. Some samples from the datasets are presented in Figures 4 and 5. The datasets are annotated with a list of point pairs, representing line segments.

York dataset was annotated under so-called Manhattan world assumption [29], which means that the annotated line segments are those aligned with the basis of some Cartesian coordinate system (specifically, with axes parallel to image sides), while the others are ignored. Wireframe dataset did not follow Manhattan world assumption and was annotated with line segments, from which "meaningful geometric information of the scene can be extracted" [23], which also resulted in some salient line segments being not annotated.

So, the ground truth labeling in these datasets is explicitly limited to some category of line segments - which means other categories of line segments are viewed as negatives. CNNs that are trained on these datasets (and/or with high accuracy on them) will have to systematically classify these categories of line segments as negatives and therefore - by design! - can't be viewed as general-purpose line detectors.

While such annotations can be useful to train and test some specific niche line segment detectors (e.g. for indoor robot navigation [30]) we believe they are flawed as datasets for general purpose line segment detection. Specifically, we would like to highlight the following problems (also illustrated in Fig. 4b).

One problem is the inconsistency of the annotations - it is easily noticeable on strip-like objects having two side line segments looking almost exactly alike - however one of them is annotated while the other is not.

Some categories of salient line segments are systematically not annotated - e.g. shadows and reflections. We believe the fact that these segments are not "real" physical objects should not be considered in the context of general purpose line segment detection and these segments should be annotated as well.

Finally, some line segments lying on the same straight line are falsely merged (vertical segments on the bed canopy's frame in Fig. 4b). It happens when a long line segment is intercepted by another object. Although for some applications it could be desirable to avoid such a splitting and there are approaches to achieve that [12], [13], we believe, that for general-purpose detector splitting is the desired detector's behaviour.

B. WIREFRAME-TINY++

To approach the covered issues with the existing datasets, we selected 20 random images from Wireframe test subset and reannotated them to make the annotations more accurate and consistent. We call the selected subset of images with the original markup Wireframe-tiny, and the resulting dataset with enhanced annotations - Wireframe-tiny++.

Comparing to the original annotations, we mainly added unannotated segments, 9 per image on average. Some segments are removed as undetectable. Some segments are divided into several smaller segments due to occlusion. The refined annotations are presented in Fig. 4c.

V. EXPERIMENTAL SETTING

A. DATASETS

The proposed algorithm is trained and evaluated with the following datasets. Wireframe dataset [23] consisting of 5000 training and 462 test images is used both to train and evaluate LSDNet. Datasets YorkUrban [24], Wireframe-tiny and its reannotated version Wireframe-tiny++ (refer to the previous section for details), composed of 120 and 20 images correspondingly, are used solely for evaluation.

B. ACCURACY

To evaluate LSDNet accuracy we use standard [15], [18] quality score $F^H = 100 \cdot 2 \cdot p \cdot r/(p + r)$, where p, r stand for precision and recall. Multiplier 100 is added for readability, making F^H fall in [0, 100] range. The score is evaluated pixel-wise by rasterizing both the predicted and the reference line segments. A pixel of a predicted line segment is considered true positive, if its distance to a pixel of a reference segment does not exceed 1% of image diagonal. For evaluation we use F^H implementation provided with L-CNN [18].

Quality score F^H was criticised [18] for being not sensitive towards overlapping and splitted line segments. We consider such an insensibility is not critical: LSDNet can not produce overlapping segments by design, since line support regions (LSRs) can not overlap, and we did not observe a notable amount of splitted line segments for any CNN-based algorithm.

C. SPEED

CNN is benchmarked on Quadro GV100 GPU for comparison with other detectors. Reconstruction algorithm is benchmarked on Core i5 9300hf CPU. CNN and the reconstruction algorithm are benchmarked independently. The speed of the latter depends on its input, we report the average speed over the dataset given the trained preprocessing network.

D. BASELINES

We compare the proposed LSDNet with classical algorithm LSD [8] and several state-of-the-art CNN-based detectors L-CNN [18], HAWP [15], TP-LSD [17], M-LSD and M-LSD-tiny [20].

The reported FPS for all CNN-based methods is cited as in [20], where benchmarking was performed on Tesla V100 GPU with practically the same characteristics as the GPU used in our experiments.

The reported F^H is also cited as in [20] for all methods except LSD and L-CNN [18], for which it was reproduced by our means. We tried to reproduce the stated quality measurements for other approaches with the help of their open-source implementations, but they appeared notably lower than the reported ones. Therefore on datasets Wireframe-tiny and Wireframe-tiny++ we compare LSDNet only to L-CNN and LSD.

E. PREPROCESSING

For LSDNet, all images are resized to 288 × 288, which appeared to be the optimal input shape in terms of speed-accuracy tradeoff. Pixel intensities are simply converted from 8-bit unsigned integer to 32-bit floating-point

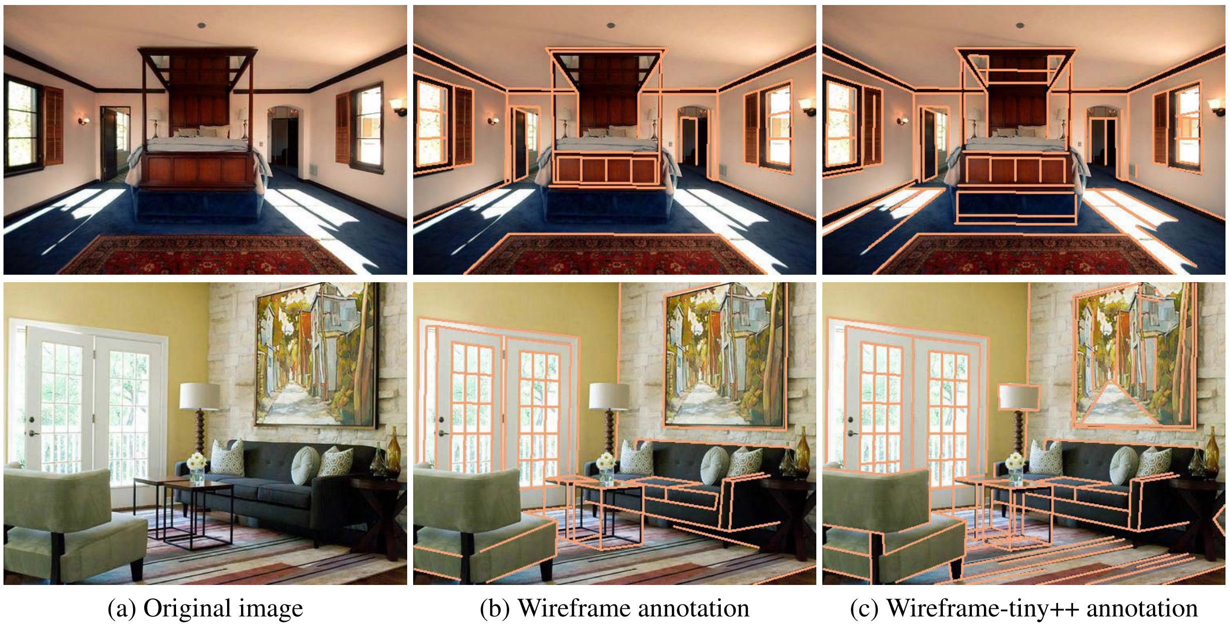

FIGURE 4. An image from Wireframe dataset (a) and two versions of its annotation - original (b) and proposed (c). Best viewed in color and zoom-in.

with 1/255 scaling coefficient. During training, random horizontal and vertical flips and gamma correction are applied.

For baseline methods, the preprocessing from the corresponding paper is applied. For LSD, we use 320 × 320 image shape.

F. HYPERPARAMETERS

LSDNet is initialized with He uniform initialization [31] and trained by Adam optimizer [32] with 10^{-4} weight decay and 8 images per batch for 180 epochs. The initial learning rate is 10^{-3} and is reduced by half if the value of loss function does not improve for 15 epochs.

G. IMPLEMENTATION

CNN training and inference is implemented in TensorFlow [33] and ONNX Runtime [34], correspondingly. Reconstruction algorithm is implemented in C++ with the help of OpenCV [35].

VI. RESULTS AND ANALYSIS

In this section we quantitatively and qualitatively analyze LSDNet performance and compare it to a wide range of state-of-the-art line segment detectors. Please refer to Section V for evaluation and comparison details.

A. PUBLIC DATASETS

Table 1 and Figure 5 summarize the results on Wireframe and York Urban datasets. It shows that LSDNet achieves state-of-the-art inference speed of 695 FPS for standalone CNN and 214 FPS for overall detector alongside competitive accuracy with 77.5 F^H and 64.6 F^H on Wireframe and York Urban datasets, correspondingly.

TABLE 1. Detection accuracy on datasets Wireframe and York Urban. Bold and underlined values stand for top-1 and top-2, correspondingly. Column "postproc" shows the speed of postprocessing the CNN's prediction.

Method	accuracy, F^H		speed, FPS		
	Wireframe	York	CNN	postproc	total
LSD [8]	63.3	59.0	-	185	185
L-CNN [18]	81.2	65.4	55	24	17
HAWP [15]	**83.1**	<u>66.3</u>	55	82	33
TP-LSD [17]	<u>82.0</u>	**67.3**	65	201	49
M-LSD [20]	80.0	64.2	132	<u>883</u>	115
M-LSD-tiny [20]	76.8	61.9	<u>241</u>	**1203**	<u>200</u>
LSDNet (proposed)	77.5	64.6	**695**	301	**214**

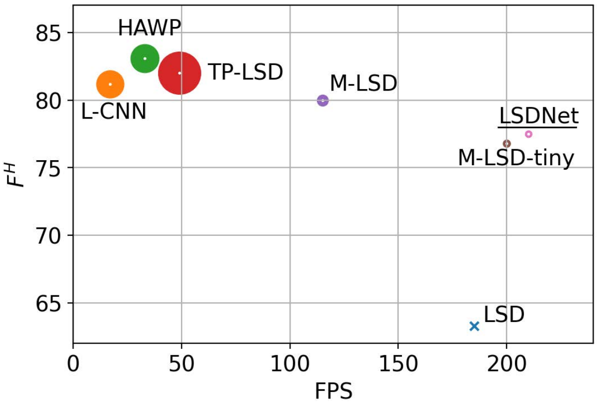

FIGURE 5. Speed (FPS) and accuracy (F^H) (accuracy) comparisons of different line segment detectors. Size of circles indicates the number of trainable parameters. LSD is marked as cross since it has no trainable parameters. The proposed LSDNet outperforms all previous detectors in speed with 214 FPS and outperforms the nearest fastest counterpart, M-LSD-tiny, both in speed and F^H quality.

In comparison with the nearest fastest counterpart, M-LSD-tiny, LSDNet outperforms it both in quality and

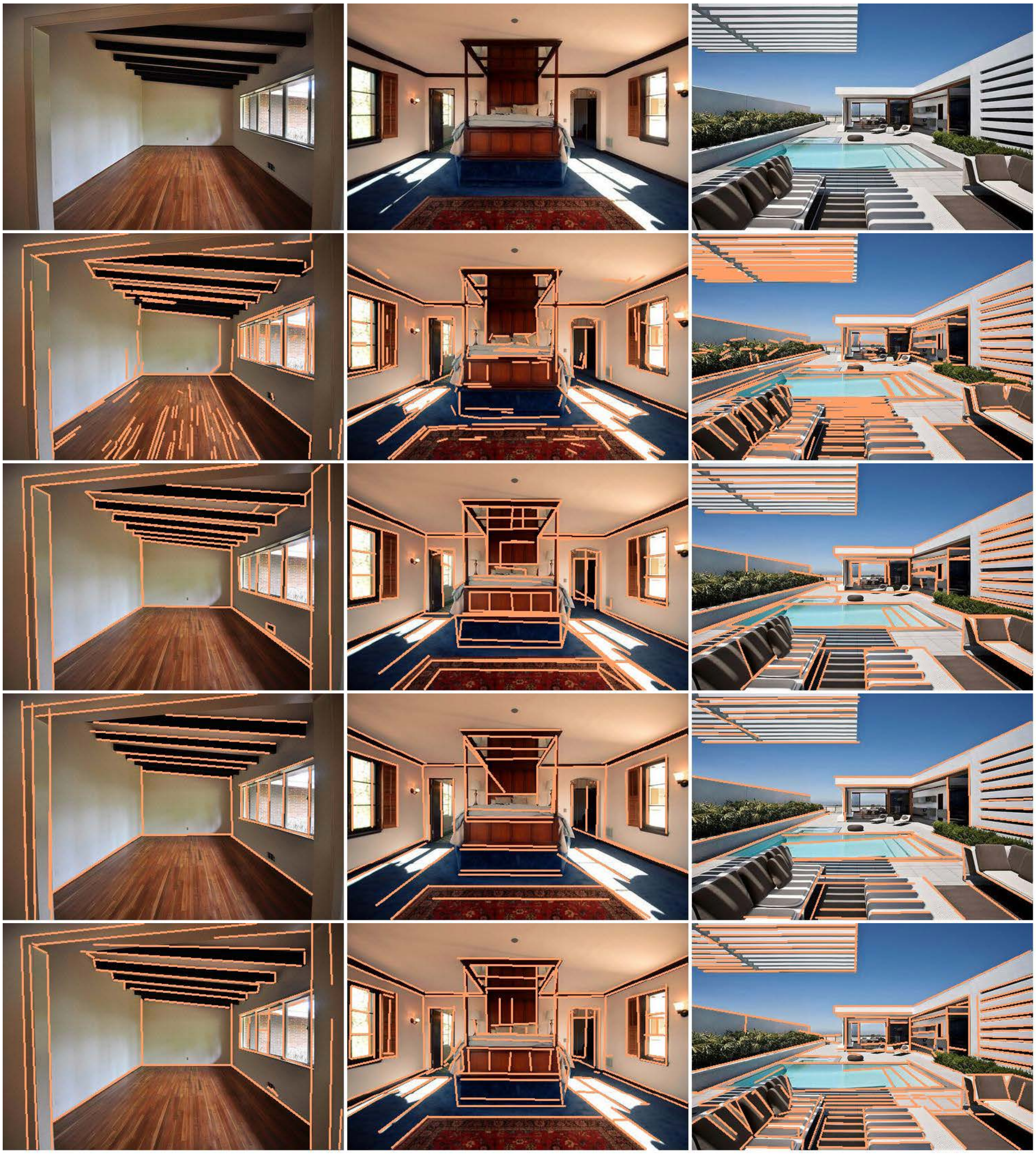

FIGURE 6. Qualitative evaluation. From left to right - images from Wireframe dataset of varying number of salient line segments. From top to bottom - original image, detection results from LSD [8], the model-based detector; HAWP [15], the most accurate CNN-based detector; M-LSD-tiny [20], the fastest CNN-based detector except for the proposed one; the proposed LSDNet. Best viewed in color and zoom-in.

speed with the absolute increase in accuracy of $+1.0\ F^H$ on Wireframe, $+2.7\ F^H$ on York Urban and +14 FPS speed-up.

Compared to the fastest detector outperforming LSDNet in accuracy on Wireframe dataset, M-LSD, the proposed approach is approximately two times faster with 214 FPS against 115 FPS. The fastest algorithm to outperform LSDNet on both datasets is TP-LSD, which is approximately four times slower with 49 FPS.

TABLE 2. Detection accuracy on datasets Wireframe, Wireframe-tiny, Wireframe-tiny++ datasets. Bold and underlined values stand for top-1 and top-2, correspondingly.

Method	detection accuracy, F^H		
	Wireframe	Wireframe-tiny	Wireframe-tiny++
LSD [8]	63.3	69.0	73.2
L-CNN [18]	**81.2**	**83.6**	**85.5**
LSDNet (proposed)	<u>77.5</u>	<u>82.0</u>	<u>85.3</u>

B. CUSTOM DATASETS

Table 2 summarizes the results on Wireframe dataset, its subset Wireframe-tiny and its reannotated version Wireframe-tiny++. Please refer to section IV-B for details.

On Wireframe-tiny++, all the detectors demonstrate higher accuracy than those on Wireframe-tiny. Since these datasets are composed of the same images and differ only in annotations, such a consistent accuracy growth indicates that Wireframe-tiny++ annotation is more suitable for the problem of general purpose line segment detection.

All the approaches, being arranged by F^H, show the same relative order on all the datasets, but the absolute differences change significantly. The gap between LSDNet and L-CNN has shrinked from 3.7 F^H on Wireframe to negligible 0.2 F^H on Wireframe-tiny++. We believe it could be explained by different learning capacity of detectors' CNNs. Expressive L-CNN with 9.8M parameters managed to learn the subtle notion of a line segment implied by Wireframe train dataset annotation (discussed in Sec. IV-A); whereas lightweight LSDNet with only 0.5M parameters learned the general line segment detection with no capacity to learn the subtle details. It made L-CNN good for wireframe-like detection problems with the goal to detect line segments, from which "meaningful geometric information of the scene can be extracted". But it could possess confusing properties in terms of general-purpose line segment detection, making LSDNet a better choice in such a case.

C. QUALITATIVE RESULTS

Qualitative comparison of LSDNet to other line segment detectors is illustrated in Figure 6. In this section we refer to LSD and LSDNet as LSR-based and to HAWP and M-LSD-tiny as endpoint-based detectors, since the methods within these groups demonstrate similar behaviour.

Endpoint-based detectors demonstrate the selectivity of line segments, which could not be attributed to overall segments' saliency. This effect is mostly notable in the foreground in the right column in Figure 6. LSR-based LSD detects all the shadows and the carpet in front of the sofa, while it can't "see" floor tiles due to their low contrast. LSDNet detects all the shadows, the carpet and the floor tiles. Whereas endpoint-based detectors HAWP and M-LSD-tiny detect these objects poorly, but at the same time they detect way less salient segments in the background. We believe such a selectivity could be attributed to the combination of high expressive power of the underlying CNN and annotation inconsistencies of train dataset, discussed in Section IV-A. This effect could be undesirable in an application requiring that very class of segments, which is missed by endpoint-based detectors.

Another interesting difference between the detectors' groups occurs due to the types of misdetections. In terms of a quality measure, LSR-based LSD and LSDNet can detect false positives, typically corresponding to line segment-like patterns on highly structured image regions, and miss some annotated segments (false negatives), which are usually poorly visible. These errors could be, at least partially, attributed to the ill-posed nature of the task. The endpoints-based methods are also prone to miss poorly visible segments, and possess an advantage of not detecting false positives on structured image regions. However, an potential drawback of endpoints-based detectors is that they can produce hard false positives — line segments of high confidence score with no evidence of a true line segment in an image. We believe the reason for hard false positives is a classification error of line segment verification module. An example could be seen in the middle column in Figure 6, please note the salient diagonal segments in the bedhead (fourth row) and right part of the carpet (third row). This issue is to be approached prior to successful exploitation of an endpoint-based detector.

VII. CONCLUSION

In this study we introduce a fast and accurate line segment detector LSDNet. The detector is composed of a lightweight encoder-decoder CNN, which predicts line segment heatmap and tangent field, and a postprocessing module – a modification of the famous LSD algorithm. When benchmarked on the traditional Wireframe dataset against several SOTA methods, LSDNet shows the highest FPS of 214 – though it achieves detection accuracy of 78 F^H – lower than the best methods (82 and 83.1 F^H).

However, we speculate that this gap in detection accuracy is primarily caused by the imperfections of the dataset rather than the network itself. We analyze the commonly used line segment detection datasets – Wireframe and York Urban – and point out numerous and significant inconsistencies in their annotation. By carefully reannotating a part of the Wireframe test dataset, we show that (i) all detectors demonstrate better quality on improved annotations (without any re-training), which indicates that the refined annotations correlate better with the notion of correct line segment detection, (ii) the gap between accuracies of our detector and others is reduced to almost non-existent - with our method being the fastest.

ACKNOWLEDGMENT

The authors would like to thank Marina Tepliakova for making the illustrations; Alexey Savchik and Veniamin Blinov for reviewing the early versions of the manuscript; and Dmitry Nikolaev for his strong belief in the power of fusing model-based algorithms with light-weight neural networks, which inspired this work.

REFERENCES

[1] D. C. Andrade, F. Bueno, F. R. Franco, R. A. Silva, J. H. Z. Neme, E. Margraf, W. T. Omoto, F. A. Farinelli, A. M. Tusset, S. Okida, M. M. D. Santos, A. Ventura, S. Carvalho, and R. D. S. Amaral, "A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 1497–1507, Apr. 2019.

[2] T. M. Hoang, H. Hong, H. Vokhidov, and K. R. Park, "Road lane detection by discriminating dashed and solid road lanes using a visible light camera sensor," *Sensors*, vol. 16, no. 8, p. 1313, 2016.

[3] O. Shipitko, V. Kibalov, and M. Abramov, "Linear features observation model for autonomous vehicle localization," in *Proc. 16th Int. Conf. Control, Automat., Robot. Vis. (ICARCV)*, 2020, pp. 1360–1365.

[4] A. Ceron, B. I. F. Mondragon, and F. Prieto, "Power line detection using a circle based search with UAV images," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2014, pp. 632–639.

[5] Y. Cao and L. Yan, "Automatic road network extraction from UAV image in mountain area," in *Proc. 5th Int. Congr. Image Signal Process.*, Oct. 2012, pp. 1024–1028.

[6] J. Shemiakina, A. Zhukovsky, and D. Nikolaev, "The method for homography estimation between two planes based on lines and points," *Proc. SPIE*, vol. 10696, Apr. 2018, Art. no. 106961G.

[7] A. Zhukovsky, D. Nikolaev, V. Arlazarov, V. Postnikov, D. Polevoy, N. Skoryukina, T. Chernov, J. Shemiakina, A. Mukovozov, I. Konovalenko, and M. Povolotsky, "Segments graph-based approach for document capture in a smartphone video stream," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, Nov. 2017, pp. 337–342.

[8] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.

[9] C. Akinlar and C. Topal, "Edlines: Real-time line segment detection by edge drawing (ed)," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 2837–2840.

[10] R. F. C. Guerreiro and P. M. Q. Aguiar, "Connectivity-enforcing Hough transform for the robust extraction of line segments," *IEEE Trans. Image Process.*, vol. 21, no. 12, pp. 4819–4829, Dec. 2012.

[11] Z. Xu, B.-S. Shin, and R. Klette, "Accurate and robust line segment extraction using minimum entropy with Hough transform," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 813–822, Mar. 2015.

[12] N. Hamid and N. Khan, "LSM: Perceptually accurate line segment merging," *J. Electron. Imag.*, vol. 25, no. 6, Dec. 2016, Art. no. 061620.

[13] Q. Yu, G. Xu, Y. Cheng, and Z. H. Zhu, "PLSD: A perceptually accurate line segment detection approach," *IEEE Access*, vol. 8, pp. 42595–42607, 2020.

[14] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, "Learning attraction field representation for robust line segment detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1595–1603.

[15] N. Xue, T. Wu, S. Bai, F. Wang, G.-S. Xia, L. Zhang, and P. H. S. Torr, "Holistically-attracted wireframe parsing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2788–2797.

[16] Y. Lin, S. L. Pintea, and J. C. van Gemert, "Deep Hough-transform line priors," 2020, *arXiv:2007.09493*.

[17] S. Huang, F. Qin, P. Xiong, N. Ding, Y. He, and X. Liu, "TP-LSD: Tri-points based line segment detector," 2020, *arXiv:2009.05505*.

[18] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 962–971.

[19] K. Huang and S. Gao, "Wireframe parsing with guidance of distance map," *IEEE Access*, vol. 7, pp. 141036–141044, 2019.

[20] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin, "Towards real-time and light-weight line segment detection," 2021, *arXiv:2106.00186*.

[21] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[22] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9627–9636.

[23] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 626–635.

[24] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating Manhattan frames in urban imagery," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2008, pp. 197–210.

[25] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 4, pp. 425–455, Jul. 1986.

[26] L. Teplyakov, K. Kaymakov, E. Shvets, and D. Nikolaev, "Line detection via a lightweight CNN with a Hough layer," 2020, *arXiv:2008.08884*.

[27] Z. Li, F. Wang, and N. Wang, "LiDAR R-CNN: An efficient and universal 3D object detector," 2021, *arXiv:2103.15297*.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Springer, 2015, pp. 234–241.

[29] J. M. Coughlan and A. L. Yuille, "Manhattan world: Orientation and outlier detection by Bayesian inference," *Neural Comput.*, vol. 15, no. 5, pp. 1063–1088, May 2003.

[30] E. Delage, H. Lee, and A. Y. Ng, "Automatic single-image 3D reconstructions of indoor Manhattan world scenes," in *Robotics Research*. Springer, 2007, pp. 305–321.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[33] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[34] *Optimize and Accelerate Machine Learning Inferencing and Training*. [Online]. Available: https://www.onnxruntime.ai/

[35] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 120–123, 2000. [Online]. Available: https://opencv.org/

LEV TEPLYAKOV received the B.S. and M.S. degrees from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Institute for Information Transmission Problems, Moscow.

His master's thesis considered the problem of training cascade CNN classifiers, which process an image with fast or slow branches, depending on the complexity of an image. His research interests include video analytics, action recognition, deep learning for object detection, and fast neural networks.

LEONID ERLYGIN received the bachelor's degree from the Moscow Institute of Physics and Technology, in 2021. He is currently pursuing the master's degree with the Institute for Information Transmission Problems, Moscow, Russia.

His research interests include CNN-based object detection and classification, with the focus on generation of synthetic training datasets with the help of data augmentation and 3-D modeling.

EVGENY SHVETS graduated from the Moscow Institute of Physics and Technology, Moscow, Russia, and received the Ph.D. degree in technology from the Institute for Information Transmission Problems, Moscow, in 2017.

His Ph.D. thesis focused on distributed control of a swarm for area surveillance. His research interests include image processing, image registration, and deep learning, including generation of synthetic datasets for neural networks training.