Semantic Line Detection and Its Applications

Jun-Tae Lee, Han-Ul Kim
Korea University
jtlee,hanulkim@mcl.korea.ac.kr

Chul Lee
Pukyong National University
chullee@pknu.ac.kr

Chang-Su Kim
Korea University
changsukim@korea.ac.kr

Abstract

Semantic lines characterize the layout of an image. Despite their importance in image analysis and scene understanding, there is no reliable research for semantic line detection. In this paper, we propose a semantic line detector using a convolutional neural network with multi-task learning, by regarding the line detection as a combination of classification and regression tasks. We use convolution and max-pooling layers to obtain multi-scale feature maps for an input image. Then, we develop the line pooling layer to extract a feature vector for each candidate line from the feature maps. Next, we feed the feature vector into the parallel classification and regression layers. The classification layer decides whether the line candidate is semant ic or not. In case of a semantic line, the regression layer determines the offset for refining the line location. Experimental results show that the proposed detector extracts semantic lines accurately and reliably. Moreover, we demonstrate that the proposed detector can be used successfully in three applications: horizon estimation, composition enhancement, and image simplification.

1. Introduction

The layout of an image is characterized by significant lines in many cases [11, 14]. For example, in Figure 1, a horizontal line in a static scene makes viewers comfortable, while diagonal lines bring a sense of dynamics. Also, a mirrored scene across a symmetric line looks balanced, and vertical lines convey an uplifting feeling. Such significant lines, separating different semantic regions in a scene, are called *semantic lines*. Note that, rather than being obvious line segments, semantic lines are often *implied* by boundaries between semantic regions, as in Figures 1(a) and (b).

Knowing the arrangement of semantic lines is important in understanding images. However, little work has been made for detecting semantic lines, whereas various techniques have been proposed to detect main subjects or points, including salient object detection [8, 37, 42] and vanishing point detection [40, 41]. Also, horizon detection schemes have been proposed in [23, 38, 41], but horizons compose only a small subset of semantic lines. In this work, we first propose an automatic algorithm to detect a wide variety of semantic lines.

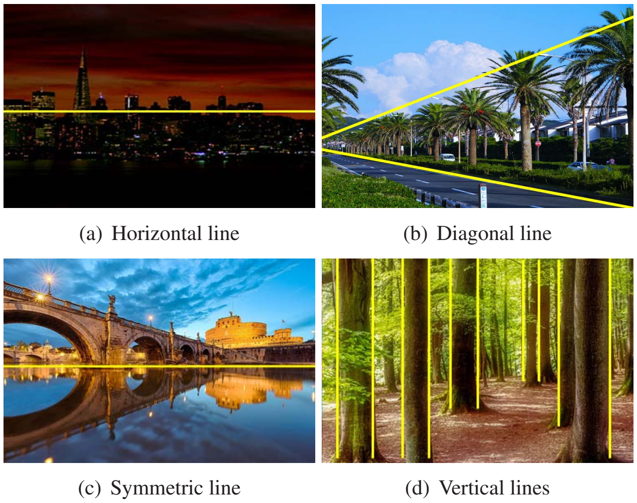
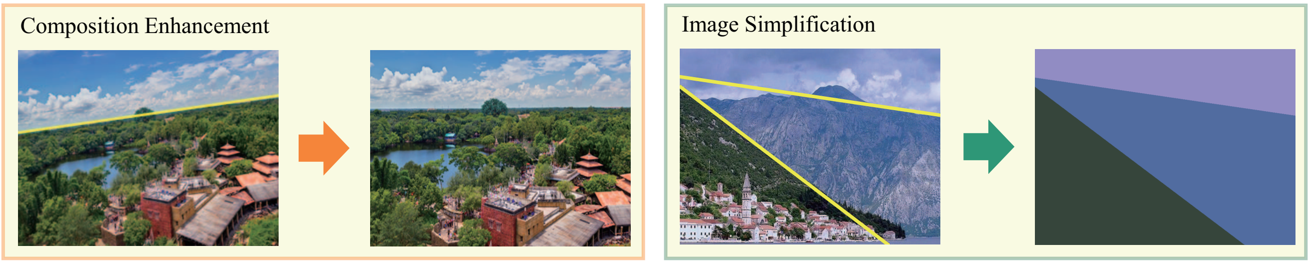

(a) Horizontal line
(b) Diagonal line
(c) Symmetric line
(d) Vertical lines

Figure 1. Semantic line examples.

Semantic line detection has practical importance, since it is applicable to various computer vision tasks. For instance, semantic lines can be used to estimate the levelness of an image, which is one of the important factors to enhance photographic composition [14, 21] in image aesthetics [22, 27]. It is difficult for amateurs to adjust the levelness. Semantic line detection can facilitate this task, as in Figure 2(a). Note that there are several methods for estimating the levelness of images [13, 23]. Koo and Nam [23] detect skewed horizons. Fischer *et al.* [13] estimate a skew angle of a rotated image using a neural network. However, it is hard to enhance image composition using their method, which does not provide explicit semantic cues, such as horizons. In [24, 28], skew estimation methods, tailored for document images, also have been proposed.

Moreover, notice that semantic lines are placed on boundaries of semantically important regions. Thus, we can divide an image into semantic regions along semantic lines and simplify the image by discarding color and texture details in each region and retaining only the essential information, as shown in Figure 2(b). Such a simplified

2380-7504/17 $31.00 © 2017 IEEE
DOI 10.1109/ICCV.2017.350



(a) (b)

Figure 2. Applications of semantic line detection: detected semantic lines are used to enhance the levelness of an image in (a) or to simplify an image by discarding color and texture details and retaining only the spatial layout in (b).

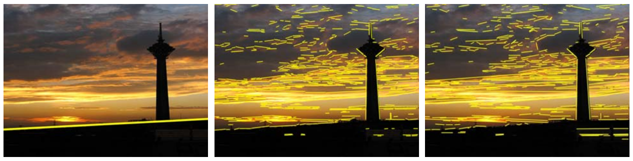
image can be exploited as a useful prior in many vision applications, including single image depth estimation [10,34], texture transfer [15], and semantic scene segmentation and understanding [16,33].

Despite the importance of semantic lines, it is not straightforward to detect them. While some semantic lines are apparent and clearly identifiable, others are less obvious and should be inferred from various cues. For example, in Figure 3(a), a semantic line is detected based on a holistic understanding of the image to separate the sky from the ground. Note that it is different from line segments in Figures 3(b) and (c), extracted by conventional line segment detectors [4, 19]. A lot of fragmented linear edges are detected across which gray levels or colors vary rapidly.

In this work, we propose a semantic line detector, based on a convolutional neural network (CNN), called *semantic line network* (SLNet). Note that CNNs can extract high level features, which are essential for understanding image semantics, more effectively than the traditional feature engineering [36]. Inspired by recent successes in multi-task learning [17, 26, 43], we regard semantic line detection as a combination of two tasks: classification and regression. More specifically, the proposed SLNet first uses a series of convolution layers to yield feature maps of an input image. Then, two line pooling layers extract the line features for a candidate line from feature maps at different scales, which are then concatenated and fed into the parallel classification and regression layers. The classification layer determines whether a candidate line is semantic or not. In case of a semantic line, the regression layer refines its location by computing the regression offset. Experimental results show that the proposed SLNet detects implied, as well as obvious, lines accurately and reliably.

We make the following major contributions:

- We propose the notion of semantic line and construct a publicly available dataset for semantic line detection, called the semantic line (SEL) dataset, in which each image is annotated with ground-truth semantic lines.
- We propose the semantic line detector, by employing a CNN with multi-task, multi-scale learning. We also develop the line pooling layer to extract the features of semantic lines effectively.



(a) (b) (c)

Figure 3. Semantic lines are different from line segments. A semantic line is detected by the proposed algorithm in (a), whereas line segments are extracted by the LSD method [19] in (b) and the EDLines method [4] in (c).

- We provide two semantic line detection modes for various applications: primary semantic line detection and multiple semantic line detection. To demonstrate its many potential applications, we apply the proposed semantic line detector to horizon estimation, composition enhancement, and image simplification.

2. Related Work

2.1. Local Region Representation in CNNs

Since CNNs can extract high level, as well as low level, features effectively, they have been adopted in various vision applications [17, 18, 30] recently. In particular, object detection [17, 32] and semantic segmentation [9, 20] need to analyze local regions, rather than an entire image. One approach is to crop local regions from an image and feed them into a CNN [18, 29]. However, different local regions may involve the same computations. To save such redundant computations, attempts have been made to develop a CNN that extracts regional features from the convolutional feature map for a full image [7,9,17]. Dai *et al.* [9] proposed the convolutional feature masking, in which the activations of a feature map corresponding to a target region are kept and the others are set to 0. In [7, 17], the max-pooling is applied on a target region. Whereas Girshick [17] performed the pooling on the tight bounding box of a target region, Caesar *et al.* [7] developed the free-form max-pooling.

2.2. Multi-Task Learning

Multi-task learning refers to the joint training of a network to solve multiple problems, which share common lay-

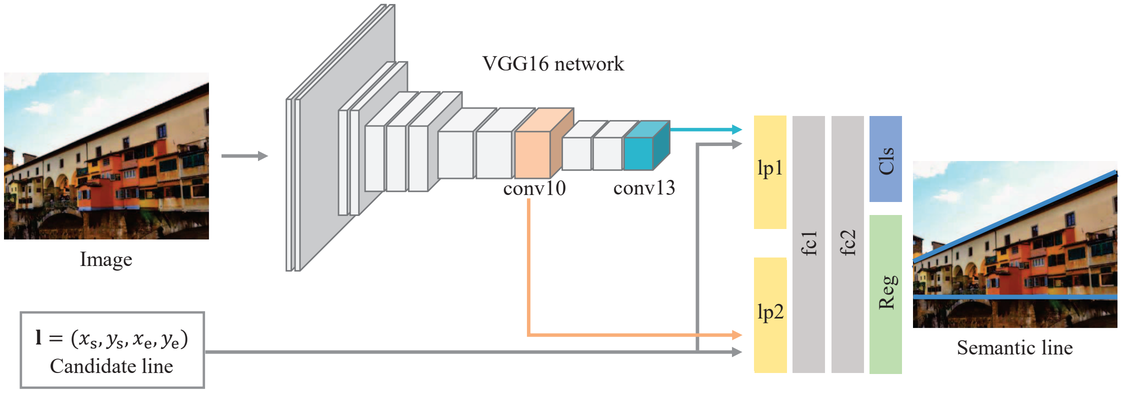


Figure 4. The architecture of SLNet, which contains 13 convolution layers (conv1 ∼ conv13), two line pooling layers (lp1 and lp2), and two fully connected layers (fc1 and fc2). 'Cls' and 'Reg' denote the classification and regression layers, respectively.

ers for intermediate parameterization [31]. If the problems are sufficiently related, multi-task learning can lead to better generalization and benefit all the problems. For instance, Girshick [17] addressed the problem of object detection as the combination of classification and localization: the classification layer categorizes an object proposal into one of the pre-defined classes, and the localization layer outputs a refinement offset of the object location through the regression. Inspired by [17], we develop a multi-task CNN to dichotomize each line candidate into either semantic or non-semantic class and, in case of a semantic line, to determine the offset for refining the line location.

2.3. Semantic Lines in Photographic Composition

Photographic composition is determined by the arrangement of subjects in a photograph. Semantic lines are important composition elements. In well-composed photographs, semantic lines direct viewers' attention to the photographers' intended focal points and yield visually balanced images [14, 21]. There are several composition rules for obtaining attractive photographs based on semantic lines. For example, a horizontal line tends to divide an image height into the golden ratio [6, 14] in a visually balanced image. Also, it is desirable to locate a slanted semantic line on an image diagonal [21,25]. In this way, semantic lines are crucial to the aesthetic qualities of images, and they provide important information for understanding the spatial layouts.

3. Proposed Semantic Line Detector

We detect semantic lines in an image by classifying and regressing candidate lines using the proposed network, SLNet. We generate candidate lines by connecting two points, which are uniformly sampled on image boundaries. We parameterize a line $\mathbf{l}$ with a quadruple (l_1, l_2, l_3, l_4), or in alternative notations $(x_\mathrm{s}, y_\mathrm{s}, x_\mathrm{e}, y_\mathrm{e})$, where $(x_\mathrm{s}, y_\mathrm{s})$ and $(x_\mathrm{e}, y_\mathrm{e})$ are the start and end points of $\mathbf{l}$, respectively.

3.1. Semantic Line Network: SLNet

Figure 4 shows the architecture of SLNet that takes an input image and a candidate line, and yields the classification and regression results. SLNet is based on the VGG16 network [36]. Specifically, we employ the 13 convolutional layers (conv1 ∼ conv13) in VGG16. Then, for subsequent layers, we design two line pooling layers (lp1 and lp2) and two fully connected layers (fc1 and fc2). The line pooling layers extract fixed-size line features from the convolutional feature maps. To take advantage of multi-scale features [5], we extract two line features from different convolutional layers (conv10 and conv13), where conv10 is twice larger than conv13. The line features are concatenated and fed into fc1 and fc2. Finally, the network branches into two parallel output layers: one for classifying the candidate line (Cls), and the other for computing regression offsets for the line parameters (Reg).

Multi-Task Loss Function: We divide the semantic line detection problem into the classification and regression tasks and conquer the two tasks using the Cls and Reg layers in Figure 4. The classification layer Cls computes the softmax probability vector $\mathbf{p} = (p, q)$, where p and q are the probabilities that a candidate line $\mathbf{l}$ belongs to the *semantic* and *non-semantic* classes, respectively. The regression layer Reg outputs a line offset $\Delta\mathbf{l} = (\Delta l_1, \Delta l_2, \Delta l_3, \Delta l_4)$. For scale-invariance, the offset $\Delta\mathbf{l}$ is normalized by the minimum of the width and the height of the input image.

A training candidate line $\mathbf{l}$ is annotated with the ground-truth binary vector $\bar{\mathbf{p}} = (\bar{p}, \bar{q})$ and the ground-truth regression offset $\Delta\bar{\mathbf{l}} = (\Delta\bar{l}_1, \Delta\bar{l}_2, \Delta\bar{l}_3, \Delta\bar{l}_4)$. We define the multi-task loss L as

$$L(\mathbf{p}, \bar{\mathbf{p}}, \Delta\mathbf{l}, \Delta\bar{\mathbf{l}}) = L_\mathrm{cls}(\mathbf{p}, \bar{\mathbf{p}}) + L_\mathrm{reg}(\Delta\mathbf{l}, \Delta\bar{\mathbf{l}}) \qquad (1)$$

where $L_\mathrm{cls}(\mathbf{p}, \bar{\mathbf{p}})$ and $L_\mathrm{reg}(\Delta\mathbf{l}, \Delta\bar{\mathbf{l}})$ are the classification loss and the regression loss, respectively.

For the classification loss, we use the cross-entropy

$$L_\mathrm{cls}(\mathbf{p}, \bar{\mathbf{p}}) = -\bar{p}\log p - \bar{q}\log q. \qquad (2)$$

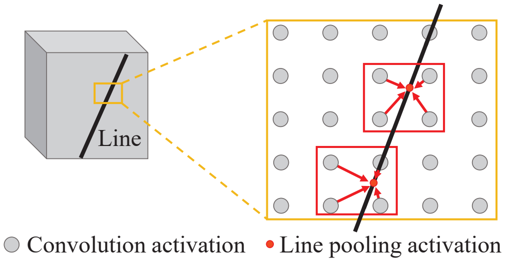


Figure 5. Illustration of the line pooling. Each line pooling activation is bilinearly interpolated from the four nearest pixels in the convolutional feature map.

Also, we define the regression loss as

$$L_{\text{reg}}(\Delta\mathbf{l}, \Delta\bar{\mathbf{l}}) = \sum_{i=1}^{4} \eta(\Delta l_i - \Delta\bar{l}_i) \tag{3}$$

where η is the smooth L_1 function that is less sensitive to outliers,

$$\eta(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{4}$$

Line Pooling Layer: The convolutional layers compute convolutional feature maps to describe the entire image. We design the line pooling layer to extract a line feature (*i.e.* regional representation of a candidate line) from a convolutional feature map. To obtain a line feature of a fixed-size M, the line pooling layer uniformly samples M pooling locations along the candidate line. However, most sampled locations are not exactly on the pixel lattice of the feature map. Thus, the line pooling layer performs bilinear interpolation on each sampled location. Let A^c_m denote the line pooling activation for channel c on the mth sampled location. Also, let B^c_n be the activation on the nth pixel in the feature map. Then, the line pooling activation is given by

$$A^c_m = \sum_{n \in \mathcal{N}(m)} w_{mn} B^c_n \tag{5}$$

where $\mathcal{N}(m)$ denotes the set of the four nearest pixels to the mth sampled location, and w_{mn} is the bilinear weight for B^c_n. Figure 5 illustrates the bilinear interpolation for computing line pooling activations.

In the backpropagation, the derivatives are routed through the line pooling layer. The backward function of the line pooling layer computes the partial derivative of the loss function in (1) with respect to each input activation via

$$\frac{\partial L}{\partial B^c_n} = \sum_{m : n \in \mathcal{N}(m)} w_{mn} \frac{\partial L}{\partial A^c_m}. \tag{6}$$

Instead of the proposed linear pooling, we may perform the conventional RoI pooling [17] over the bounding box of a candidate line. However, this box pooling may be ineffective for the classification purpose, when both semantic and non-semantic lines are within the same box and their features are blended. We will show that the proposed line pooling layer outperforms the box pooling in Section 4.

Multi-Scale Line Feature: Each convolution layer aggregates input activations into an output activation. Thus, in a later convolution layer, a single activation describes a larger region of an image. In other words, a later layer has a larger receptive field. Hence, lp1, which extracts the line feature from the last convolution layer (conv13), is effective in detecting semantic lines, implied by large-scale objects. However, because of its large receptive field, lp1 cannot locate semantic lines with a high level of precision. To overcome this problem, we use an additional line pooling layer, lp2, to extract a finer-scale line feature from conv10. Note that the receptive field of conv13 is about twice larger than that of conv10. Both line pooling layers lp1 and lp2 extract line features of a fixed-size M. We normalize these features to equalize their contributions and concatenate them to construct a multi-scale line feature.

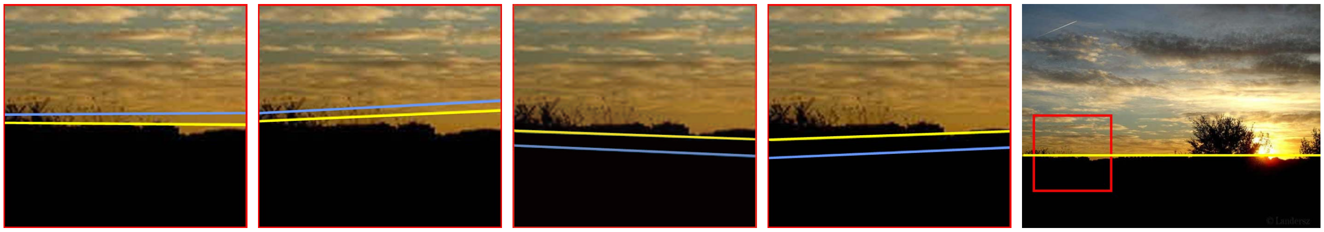
SLNet Training: We initialize the parameters of the convolutional layers using the pre-trained VGG16 network [36] on the ILSVRC-2012 dataset [35]. The other layers are initialized with zero-mean Gaussian random numbers. We train SLNet with the SEL dataset. The parameters are updated via the stochastic gradient descent with a batch size of 200 line candidates, minimizing the loss function in (1). We start with a learning rate $\epsilon = 0.001$ for all layers and shrink it via $\epsilon \leftarrow 0.1\epsilon$ after every two epochs. A momentum of 0.9 and weight decay of 0.0005 are used.

For each training image, we flip it horizontally with probability 0.5. To use SLNet in a scale-invariant manner, an input image is resized to 400×400. Only a limited number of positive training data (*i.e.* semantic lines) are available, while there are plenty of negative data. For data balancing, we generate additional semantic lines near a ground-truth line by deviating its two end points $(\bar{x}_{\text{s}}, \bar{y}_{\text{s}})$ and $(\bar{x}_{\text{e}}, \bar{y}_{\text{e}})$. Specifically, we move the endpoints either horizontally or vertically, depending on the orientation of the line, within the range $[-20, 20]$, and use them as additional positive data with the corresponding offsets $\Delta\bar{\mathbf{l}}$'s. On the other hand, the regression offsets of all negative data are set to $\Delta\bar{\mathbf{l}} = (0, 0, 0, 0)$.

3.2. Two Semantic Line Detection Modes

For versatile use of SLNet in various applications, we provide two detection modes for semantic lines: primary and multiple line detection modes.

Primary Line Detection: The photographic composition of an image is often characterized by a single semantic line. Hence, in the primary line detection mode, we straightfor-



(a) (b) (c) (d) (e)

Figure 6. Primary semantic line detection. For easier comparison, (a)∼(d) show magnified views of the red window in (e). They contain different detection results, in which the initial candidates are in blue and the regressed lines are in yellow. Among them, the regressed line in (a) is selected as the primary semantic line. In (e), it is observed that the primary semantic line is precisely located on the horizon.

wardly detect the most important line using SLNet. First, we generate $1,308$ candidate lines in an image. Then, the classification layer computes the soft-max probability p for each candidate line. A candidate with a higher soft-max probability is regarded as semantically more important. Accordingly, we detect the candidate $\mathbf{l}^{\max}$ with the maximum soft-max probability $p^{\max}$ and obtain the regressed line $\tilde{\mathbf{l}}^{\max}$ by

$$\tilde{\mathbf{l}}^{\max} = \mathbf{l}^{\max} + \Delta\mathbf{l}^{\max} \tag{7}$$

where $\Delta\mathbf{l}^{\max}$ is the offset computed by the regression layer. Then, we declare the regressed line $\tilde{\mathbf{l}}^{\max}$ as the primary semantic line. Figure 6 shows some candidate lines, their regression results, and the selected primary semantic line.

Multiple Line Detection: An image usually contains more than one semantic lines. In the multiple line detection mode, a candidate line is declared as a semantic line and regressed similarly to (7), if its soft-max probability is higher than 0.5.

However, we carry out non-maximum suppression (NMS) to prevent multiple semantic lines from being too close to one another. To this end, assuming that a line near many edge pixels tends to be more semantic, we compute the nearby edge density for each regressed line $\tilde{\mathbf{l}}$. We first project neighboring edge pixels [39] onto $\tilde{\mathbf{l}}$. Note that the pixels, whose Euclidean distances to $\tilde{\mathbf{l}}$ are less than 3, are defined as the neighbors. Then, we define the edge density $\rho(\tilde{\mathbf{l}})$ as

$$\rho(\tilde{\mathbf{l}}) = \frac{N_{\text{edge}}}{N_{\text{all}}} \tag{8}$$

where N_{all} is the number of all pixels on the line $\tilde{\mathbf{l}}$, and N_{edge} is the number of the pixels on $\tilde{\mathbf{l}}$ to which neighboring edge pixels are projected.

Given a pair of semantic lines, we measure their distance as follows. Suppose that one line divides the image into regions R and Q, and the other into R' and Q'. We measure the intersection over union (IoU) ratios between R and R' and between Q and Q', respectively. Then, we average the two IoU ratios to compute the mean IoU (mIoU). If mIoU $>$ 0.85, we regard the lines as duplicated and suppress the line that yields a lower edge density in (8). After the NMS, we output the remaining lines as the multiple semantic lines.

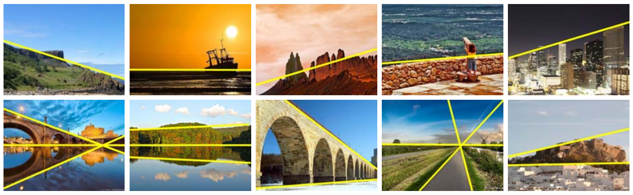


Figure 7. Sample images from the SEL dataset with the ground-truth semantic lines.

4. Experimental Results

4.1. SEL Dataset

We constructed the SEL dataset, composed of 1,750 outdoor images from photo-sharing websites [1–3]. We randomly select 90% of the images for training and use the remaining images for test. To obtain ground-truth, we requested six human subjects with basic knowledge on photographic composition to draw semantic lines on each image. They often drew lines at slightly different locations. To remove duplicated lines, we grouped lines into a cluster if their mIoU was higher than 0.85. Next, we again asked the subjects to select the representative line for each cluster. If an image has a single representative line, we set the line as the ground truth primary semantic line. If an image has multiple representative lines, we requested the subjects to rank them according to their semantic importance. We then declared the line with the best rank to be the ground truth primary semantic line, and the others to be additional ground-truth semantic lines. In the dataset, 61% of the images contain multiple semantic lines. Figure 7 shows some images with the ground-truth lines. While several semantic lines are obvious, many of them are implied. We will make this dataset publicly available.

4.2. Semantic Line Detection Results

First, we qualitatively assess the performance of the proposed semantic line detector on the SEL dataset. Figure 8 shows detection results for primary and multiple semantic lines. We see that the primary semantic lines identifies the photographic composition rules of images, such as horizon, diagonal, and symmetry. The multiple semantic lines represent the spatial layouts of images efficiently. In most cases,

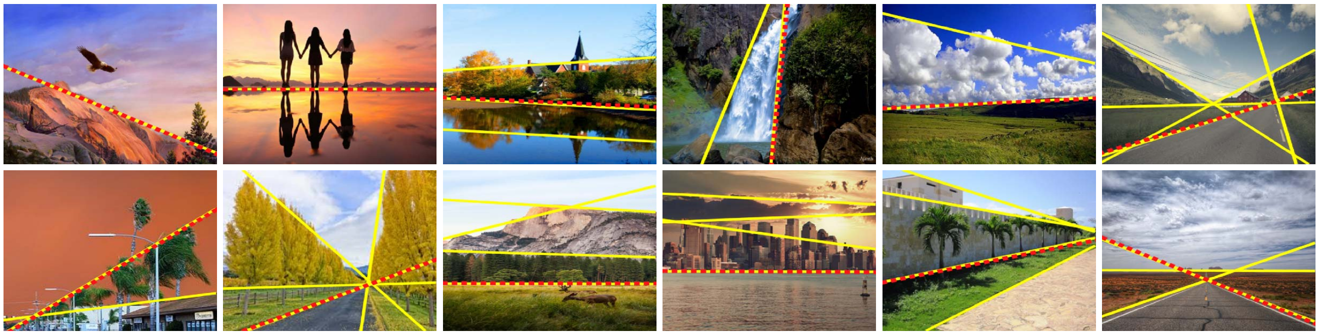


Figure 8. Semantic line detection results. Primary semantic lines are depicted by dashed red ones, while multiple semantic lines by solid yellow ones.

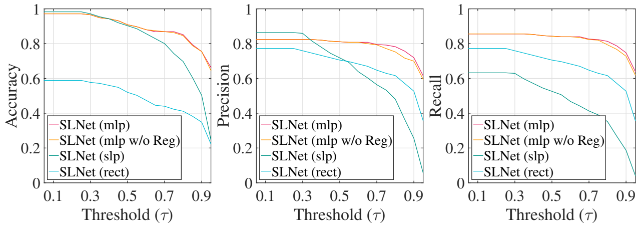


Figure 9. The accuracy, precision, and recall curves in terms of the threshold τ. In 'rect' and 'slp,' the proposed multi-scale line pooling 'mlp' is replaced by the multi-scale bounding box pooling and the single-scale line pooling, respectively. Also, 'w/o Reg' means that the regression layer is not employed.

a primary semantic line is also detected near one of the multiple semantic lines, although they are not exactly the same because of NMS in the multiple line detection mode.

Next, we evaluate the semantic line detection performance quantitatively. A semantic line is regarded as correctly detected if its mIoU score with the ground-truth line is greater than a threshold τ. Then, we define the accuracy of the primary semantic line detection as

$$\text{Accuracy} = \frac{N_c}{N} \tag{9}$$

where N_c is the number of the test images whose primary semantic lines are correctly detected, and N is the number of all test images. For the multiple semantic line detection, we define the precision and the recall as

$$\text{Precision} = \frac{N_s}{N_s + N_e}, \quad \text{Recall} = \frac{N_s}{N_s + N_m}, \tag{10}$$

where N_s is the number of correctly detected semantic lines, N_e is the number of false positives, and N_m is the number of false negatives.

Figure 9 shows the accuracy, precision, and recall curves according to the threshold τ. In Table 1, we report the area under curve (AUC) performances of the accuracy, precision, and recall curves in Figure 9, which are denoted by AUC_A, AUC_P and AUC_R, respectively. In the proposed SLNet in Figure 4, the multi-scale line pooling plays an important role in detecting semantic lines. To analyze its efficacy, instead of the line pooling layers lp1 and lp2, we adopt the RoI pooling [17] layers, in which the rectangular box pooling is performed over the bounding box of a candidate line. This scheme is denoted by 'rect' in Figure 9 and Table 1. Also, we perform the single-scale line pooling using only lp1, which is denoted by 'slp.' In addition, we test how much performance is degraded if the regression layer is not employed. This is denoted by 'w/o Reg.'

Table 1. Comparison of AUC scores (%).

	Primary	Multiple	
	AUC_A	AUC_P	AUC_R
SLNet (rect)	51.37	69.69	62.96
SLNet (slp)	84.99	67.87	50.40
SLNet (mlp w/o Reg)	90.88	79.26	82.14
SLNet (mlp)	**92.00**	**80.44**	**83.50**

In Table 1, the proposed SLNet using the multi-scale line pooling 'mlp' achieves the best detection performances for both primary and multiple semantic lines, *i.e.* the highest AUC scores 92.0%, 80.4% and 83.5% for accuracy, precision, and recall, respectively. The proposed SLNet (mlp) provides 40.6%, 10.8%, and 20.5% higher AUC_A, AUC_P, and AUC_R scores than SLNet (rect), respectively. These results indicate that the proposed line pooling layer extracts relevant features from candidate lines more effectively than the conventional RoI pooling. Moreover, SLNet (mlp) outperforms SLNet (slp) by 7.0% in AUC_A, 12.6% in AUC_P, and 33.1% in AUC_R, which indicates that the multi-scale pooling is effective for semantic line detection. Furthermore, we see that the regression layer improves the detection results, by refining the locations of detected lines.

5. Applications

We present three simple, but notable, applications of the proposed semantic line detector.

5.1. Horizon Estimation

In an image, the horizon line is defined as the projection of the line at infinity for any plane that is orthogonal to the local gravity vector [38]. Horizon lines and vanishing points provide strong characterization of geometric scene

Table 2. Comparison of AUC scores of the proposed SLNet and SLNet_HLW with the conventional methods [38,41] on the HLW dataset.

	[41]	[38]	SLNet	SLNet_HLW
AUC (%)	58.24	71.16	70.51	**82.33**

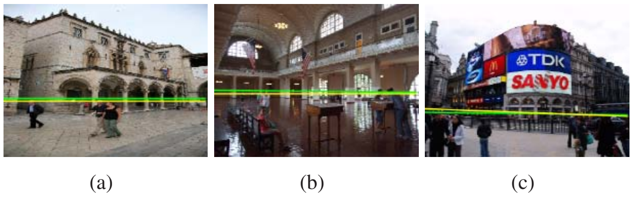

(a) (b) (c)

Figure 10. Horizon estimation results in three HLW test iamges. Green lines are ground truth horizons, and yellow lines are extracted by SLNet_HLW.

structures [41]. Thus, horizon lines are important semantic lines. We determine the horizon line in an image, by detecting the primary semantic line using the proposed SLNet.

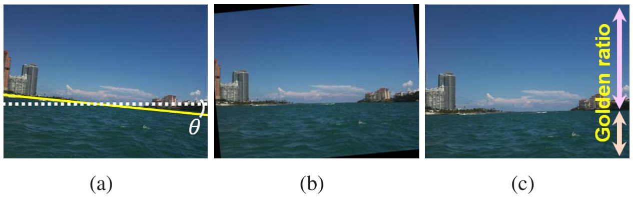
We compare SLNet with the conventional horizon estimation methods [38,41] using the HLW dataset [38], which consists of 14,000 training images and 2,018 test images. Table 2 compares the AUC scores for accuracy. Note that HLW contains many images satisfying the Manhattan world assumption, while the SEL dataset consists of mainly landscape images. However, even the original SLNet, trained with SEL, yields a comparable AUC score with the recent state-of-the-art method [38]. Moreover, SLNet trained with the HLW training images (SLNet_HLW) outperforms [38] by a large margin of 11.2%. Figure 10 shows horizon estimation results of SLNet_HLW.

5.2. Composition Enhancement

Photographic composition is an important factor in image aesthetics [22,27]. Semantic lines can be used to adjust the levelness of images and enhance their composition. If a camera is not level when taking a photograph, the output image contains a slanted horizon. Since the horizon is detected as the primary semantic line in most cases, we can use the proposed semantic line detector to adjust the horizon. As in Figure 11(a), we first obtain the slant angle θ of the detected primary semantic line. Then, we rotate the image by $-\theta$. Finally, we crop the rotated image based on two criteria: 1) the cropped region should be divided into the golden ratio by the semantic line [14,21], and 2) the cropped region should be as large as possible. Note that the final result in Figure 11(c) is visually more attractive than the input image in Figure 11(a). Figure 12 shows more composition enhancement results.

5.3. Image Simplification

For extreme image simplification, we divide an image into polygonal regions along multiple semantic lines and then make each region homogeneous, while retaining the spatial layout, as in Figure 13. Multiple semantic lines,



(a) (b) (c)

Figure 11. Composition enhancement based on the horizon adjustment: (a) input image with the detected semantic line in yellow color, (b) horizon-adjusted image, and (c) final cropping result.

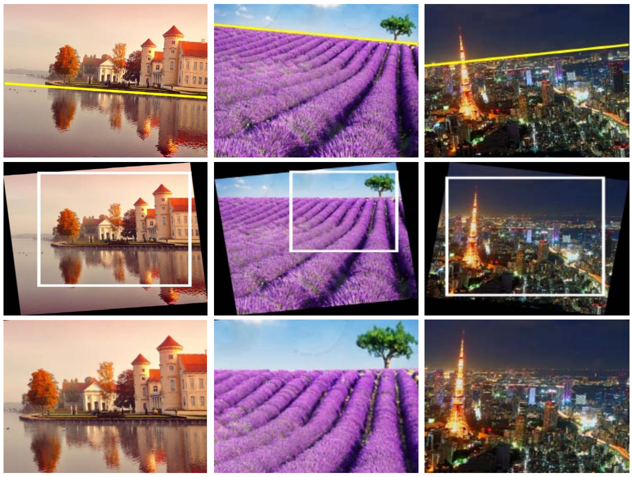


Figure 12. Composition enhancement results: (Top) original images with the primary semantic lines. (Middle) horizon-adjusted images, in which white rectangles are the regions to be cropped. (Bottom) final cropping results.

starting and ending at an image boundary, may cross one another and lead to over-segmentation, as in Figure 13(a). Hence, when detected semantic lines generate line segments, we remove some of the segments and retain the others to divide the image into polygonal regions, as in Figure 13(b).

In order to select the line segments to be removed, we define the semantic score of a line segment as

$$S = S_{\mathrm{sl}} + S_{\mathrm{ct}} + S_{\mathrm{sz}}. \tag{11}$$

First, S_{sl} is the soft-max probability of the line segment computed by SLNet. Second, S_{ct} is the contextual score. Note that the two regions R and Q, divided by the line segment, should be dissimilar from each other, if the line segment is semantic. Based on this observation, we generate the bag-of-words descriptors $\mathbf{f}_R$ and $\mathbf{f}_Q$ of R and Q using 300 words in the CIELab color space [12], and then compute the contextual score

$$S_{\mathrm{ct}} = \chi^2(\mathbf{f}_R, \mathbf{f}_{R\cup Q}) + \chi^2(\mathbf{f}_Q, \mathbf{f}_{R\cup Q}) \tag{12}$$

where $\mathbf{f}_{R\cup Q}$ is the bag-of-words descriptor of the merged region $R \cup Q$, and $\chi^2(\cdot,\cdot)$ denotes the chi-squared distance. Third, S_{sz} is the size score. It is desirable that the line seg-

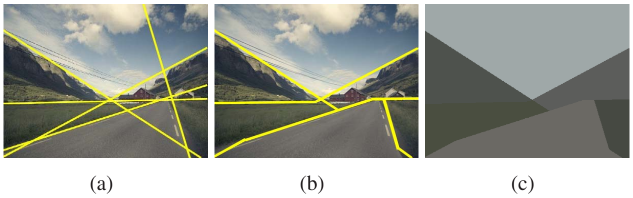


Figure 13. Image simplification: (a) detected multiple semantic lines, (b) retained line segments, (c) simplified image.

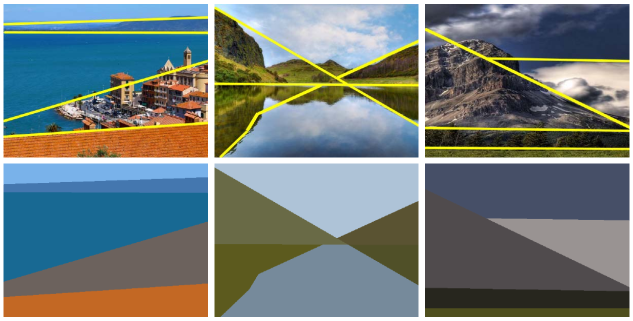


Figure 14. Image simplification: (Top) original images with retained semantic line segments. (Bottom) simplification results.

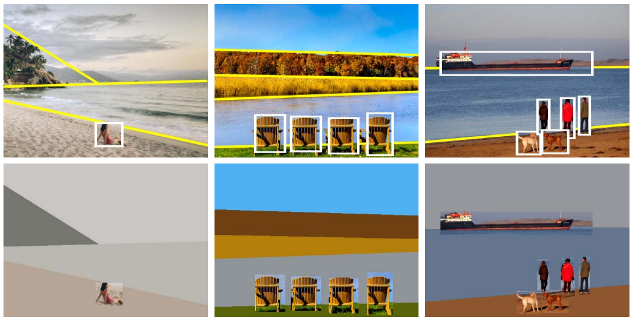


Figure 15. Image simplification with object detection: (Top) original images with retained semantic line segments and detected object boxes. (Bottom) simplification results.

ment does not yield too tiny regions. Hence, we compute

$$S_{\text{sz}} = \frac{|R \cup Q|}{A} \tag{13}$$

where $|R \cup Q|$ is the size of $R \cup Q$, and A is the size of the entire image. After computing the semantic score in (11) for each line segment, we remove the line segment with the minimum score. This is repeated until the minimum score is higher than a threshold 0.715.

Figure 14 shows simplification examples. The top row shows the retained semantic line segments. The bottom row shows simplified images, which convey the overall layouts and the essential color information. Figure 15 shows more examples, in which we employ an object detector [17] to keep object regions without simplification.

To quantify the simplification performances, we measure the mean opinion score (MOS). We requested six human subjects to assign a quality score: 1 (unacceptable), 2 (bad), 3 (medium), 4 (good), 5 (superb). Then, we averaged the six scores to obtain MOS. Also, we measure the compression ratio (CR), which equals the original raw file size (in bytes) of an image divided by the number of bytes to encode the simplification result. The parameters of a line segment require 8 bytes, and the color of a divided region needs 3 bytes. Thus, $8n_l + 3n_r$ bytes are used to encode a simplification result, where n_l and n_r are the numbers of line segments and regions, respectively. In addition, in the image simplification with object detection, each detected box is encoded by the JPEG technique.

Table 3. Average MOS and CR of image simplification results on the SEL dataset.

	MOS	CR
Simplification	3.17	8.05×10^4
Simplification with object detection	3.92	5.12×10^4

Table 3 lists the average MOS and CR performances on the SEL dataset. The simplification scheme provides MOS of 3.17 and CR of 8.05×10^4, and that with object detection yield MOS of 3.92 and CR of 5.12×10^4. These results indicate that the proposed simplification scheme can be employed as the basis of extremely low bit-rate image compression and communications. Furthermore, as mentioned before, a simplified image can be exploited in single image depth estimation [10, 34], texture transfer [15], and semantic scene segmentation and understanding [16, 33].

More application examples, as well as semantic line detection results, are available in supplemental materials.

6. Conclusions

We proposed the semantic line detector, called SLNet, and demonstrated its practical importance for higher-level vision tasks. The proposed SLNet performs the two tasks of classification and regression. It obtains multi-scale feature maps for an image. Then, the line pooling layer extracts a local feature for each line candidate from the feature maps. Next, the local feature is fed into the classification and regression layers. The classification layer decides whether the line candidate is semantic or not, and the regression layer determines the offset for locating the line more accurately. Experimental results showed that SLNet extracts implied, as well as obvious, semantic lines accurately and reliably. Furthermore, it was demonstrated that the proposed detector can be applied effectively for horizon estimation, composition enhancement, and image simplification.

Acknowledgements

This work was supported partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF2015R1A2A1A10055037), partly by the NRF grant funded by the Korea government (MSIP) (No. NRF-2016R1C1B2010319), and partly by the Agency for Defense Development (ADD) and Defense Acquisition Program Administration (DAPA) of Korea (UC160016FD).

References

[1] DPchallenge. Available: http://www.dpchallenge.com/.

[2] Flickr. Available: http://flickr.net/.

[3] Photo.net. Available: http://photo.net/.

[4] C. Akinlar and C. Topal. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, Oct. 2011.

[5] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016.

[6] S. Bhattacharya, R. Sukthankar, and M. Shah. A framework for photo-quality assessment and enhancement based on visual aesthetics. In *ACM Multimedia*, 2010.

[7] H. Caesar, J. Uijlings, and V. Ferrari. Region-based semantic segmentation with end-to-end training. In *ECCV*, 2016.

[8] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):569–582, Aug. 2015.

[9] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015.

[10] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[11] L. Excell, J. Batdorff, D. Brommer, R. Rickman, and S. Simon. *Composition: From Snapshots to Great Shots*. Peachpit Press, 2011.

[12] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[13] P. Fischer, A. Dosovitskiy, and T. Brox. Image orientation estimation with convolutional networks. In *GCPR*, 2015.

[14] M. Freeman. *The Photographer's Eye: Composition and Design for Better Digital Photos*. Focal Press, 2007.

[15] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[16] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, 2016.

[17] R. Girshick. Fast R-CNN. In *CVPR*, 2015.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[19] V. G. R. Grompone, 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.*, 32(4):722–732, Apr. 2010.

[20] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Object instance segmentation and fine-grained localization using hypercolumns. *IEEE Trans. Pattern Anal. Mach. Intell.*, PP(99):1–1, Jun. 2016.

[21] P. Jonas. *Photographic Composition Simplified.* Amphoto Publishers, 1976.

[22] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes. Photo aesthetics ranking network with attributes and content adaptation. In *ECCV*, 2016.

[23] H. I. Koo and N. I. Cho. Skew estimation of natural images based on a salient line detector. *Journal of Electronic Imaging*, 22(1):013020–013020, 2013.

[24] H. I. Koo and N. I. Cho. Robust skew estimation using straight lines in document images. *Journal of Electronic Imaging*, 25(3):033014–033014, 2016.

[25] L. Liu, R. Chen, L. Wolf, and D. Cohen-Or. Optimizing photo composition. *Comput. Graph. Forum*, 29(2):469–478, May 2010.

[26] W. Liu, T. Mei, Y. Zhang, C. Che, and J. Luo. Multi-task deep visual-semantic embedding for video thumbnail selection. In *CVPR*, 2015.

[27] L. Mai, H. Jin, and F. Liu. Composition-preserving deep photo aesthetics assessment. In *CVPR*, 2016.

[28] G. Meng, C. Pan, N. Zheng, and C. Sun. Skew estimation of document images using bagging. *IEEE Trans. Image Process.*, 19(7):1837–1846, 2010.

[29] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

[30] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *CVPR*, 2016.

[31] S. Parameswaran and K. Q. Weinberger. Large margin multi-task metric learning. In *NIPS*, 2010.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[33] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[34] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *CVPR*, 2016.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, Apr. 2015.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[37] P. Wang, J. Wang, G. Zeng, J. Feng, H. Zha, and S. Li. Salient object detection for searched web images via global saliency. In *CVPR*, 2012.

[38] S. Workman, M. Zhai, and N. Jacobs. Horizon lines in the wild. In *BMVC*, 2016.

[39] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[40] Y. Xu, S. Oh, and A. Hoogs. A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments. In *CVPR*, 2013.

[41] M. Zhai, S. Workman, and N. Jacobs. Detecting vanishing points using global image context in a non-Manhattan world. In *CVPR*, 2016.

[42] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech. Unconstrained salient object detection via proposal subset optimization. In *CVPR*, 2016.

[43] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014.