

# LCD – Line Clustering and Description for Place Recognition

Felix Taubner<sup>1</sup>, Florian Tschopp<sup>1</sup>, Tonci Novkovic<sup>1,2</sup>, Roland Siegwart<sup>1</sup>, and Fadri Furrer<sup>1,3</sup>

<sup>1</sup>ETH Zurich, Switzerland, <sup>2</sup>Gideon Brothers, <sup>3</sup>incon.ai

{ftaubner, ftschopp, rsiegwart}@ethz.ch, tonci.novkovic@gideonbros.ai, fadri@incon.ai

## Abstract

*Current research on visual place recognition mostly focuses on aggregating local visual features of an image into a single vector representation. Therefore, high-level information such as the geometric arrangement of the features is typically lost. In this paper, we introduce a novel learning-based approach to place recognition, using RGB-D cameras and line clusters as visual and geometric features. We state the place recognition problem as a problem of recognizing clusters of lines instead of individual patches, thus maintaining structural information. In our work, line clusters are defined as lines that make up individual objects, hence our place recognition approach can be understood as object recognition. 3D line segments are detected in RGB-D images using state-of-the-art techniques. We present a neural network architecture based on the attention mechanism for frame-wise line clustering. A similar neural network is used for the description of these clusters with a compact embedding of 128 floating point numbers, trained with triplet loss on training data obtained from the InteriorNet dataset. We show experiments on a large number of indoor scenes and compare our method with the bag-of-words image-retrieval approach using SIFT and SuperPoint features and the global descriptor NetVLAD. Trained only on synthetic data, our approach generalizes well to real-world data captured with Kinect sensors, while also providing information about the geometric arrangement of instances.*



Figure 1: We present a novel place recognition method that uses clusters of lines as visual and geometric features. Lines are detected in RGB-D images and clustered using a neural network. Another neural network describes the line clusters with a compact 128-dimensional embedding. These descriptors can be matched to find the corresponding scene. The colors of the lines in the images represent different clusters, and line clusters of the same color represent correct matches under different viewpoints.

## 1. Introduction

Visual place recognition is a very active area of research in computer vision and robotics [30]. In order to be able to navigate efficiently in its environment, a robot needs to know where it is and recognize scenes where it has been before. In GPS-denied areas, visual place recognition can be used as an alternative. Various current successful place recognition systems [5, 11, 3, 18] cast place recognition as an image retrieval problem. Many of them use the Bag-of-Words (BoW) approach [44, 38], which aggregate visual features such as SIFT [29] or SURF [6] to describe images with a unique representation vector. Places are recognized through similarity of visual appearance.

Recent advancements in Deep Learning (DL) and the appearance of large-scale labelled datasets have sparked numerous novel learning-based approaches for image retrieval and place recognition. Most of these use Convolutional Neural Network (CNN) features to describe images [58, 47]. While these approaches show impressive results, they typically work only with similar viewpoint directions and struggle with illumination changes.

In this work, we want to incorporate also structural cues for the place recognition task. Specifically, as clear structures are often found in human made environments, we anticipate that 3D line segments can improve place recognition. We propose to use line clusters as visual features, for the following reasons: First, lines<sup>1</sup> can be clustered to form structural

---

Code is available at <https://github.com/ethz-asl/lcd>

This work was partially supported by Siemens Mobility GmbH, Germany. We would like to thank Francesco Milano, Dominik Walder and Chengkun Li for their initial work.

©2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

---

<sup>1</sup>For brevity, the terms line segment and line are used interchangeably

instances which we assume to be less variant to illumination or viewpoint changes. Second, there are plenty of lines in human made environments. Finally, lines contain more structural information than keypoints.

Typically, visually detectable lines are object edges or texture lines. These lines can be seen from most viewing angles, while not being affected by lighting changes. In [31] and [32] the authors demonstrate the advantage of using lines as features under varying lighting conditions. In contrast, we employ a virtual camera image for description to improve viewpoint invariance.

Furthermore, lines describe object structures very well, because object contours are often made up of straight lines in human-made environments (Figure 1). We utilize the structural property of lines by describing entire clusters of line segments. What makes up a cluster is determined by the way we train the networks. A human would intuitively recognize scenes by splitting it into recognizable objects, such as a certain model of a chair, window, table, etc. Hence, we specify line clusters as the lines that belong to certain semantic instances. We define places as a conglomeration of instances that make up a scene. Thus, the task of place recognition is cast as the problem of recognizing specific instances. Instance-level visual features provide a high-level description of the place while maintaining invariance to the displacement of objects.

Our work can be divided into four main contributions that function largely independently of each other:

1. A pipeline for robust detection and reprojection of 3D line segments in an RGB-D image
2. A novel method to obtain viewpoint invariant visual descriptions of lines via virtual camera images
3. An attention-mechanism based neural network for frame-wise clustering of lines at instance level
4. A neural network for the description of line clusters with compact 128-dimensional embeddings.

## 2. Related work

Visual descriptors can be categorized as global descriptors, which describe the whole image, and local descriptors, describing local patches in the image [30]. Among the most popular hand-crafted local descriptors are SIFT, SURF and BRISK [33]. DL-based descriptors such as SuperPoint (SP) [14] have been introduced recently. Such local descriptors are typically combined using BoW [44] to create a global image descriptor. However, information on the geometric arrangement of features is lost. Alternatively, global descriptors such as Gist [36] and VLAD [4] compress the information from the whole image into a distinctive descriptor. More recent advances in image retrieval [3, 56] use neural networks for descriptor extraction. While global descriptors

---

in this paper.

are more invariant to appearance changes caused by season and lighting, they are more susceptible to variations in viewpoint [30]. With the availability of RGB-D data, point features in 3D have become more popular [42, 33]. They capture more geometric data, but are susceptible to noise and incomplete 3D data [45] which can be alleviated to some extent with learning approaches [20, 59].

In contrast to point features, lines can better handle illumination and viewpoint changes, are more robust in low textured environments with occlusions, convey structural information and provide complementary information [27, 52, 34]. Typical ways to detect lines are shown in [19, 1]. LDB [61] and MSLD [55] use line surroundings for description, and are extended for scale invariance [52] and further improved using neural networks [26]. Geometric properties of clustered nearby lines are used for description in [54].

Since global descriptors can only provide an approximate pose corresponding to the closest previously observed image, a localization approach that combines local and global visual features was proposed [41]. A city scale localization scheme is presented in [46]. Semantic information can further improve localization [49]. Using existing region proposal networks to detect landmarks of interest and a CNN based visual descriptor to identify and recognize these landmarks is presented in [47]. In addition, RGB-D images and semantic segmentation can be used to describe the structure of object arrangements [15]. 3D features [17] and later a combination of 2D and 3D features [16] enable the matching and combination of objects. The BoW approach is used in a wide variety of applications, *e.g.* in [2, 21, 24] and extended to include 3D information [8, 12, 37, 50]. [23] use MSLD and a vocabulary tree to do Simultaneous Localization And Mapping (SLAM). [34] use line segments to localize in a map by matching to a virtual view. [48] propose place recognition using line pairs and BoW, showing better performance than ORB. The advantage of using line segments as visual features has been demonstrated for SLAM [60, 31] and place recognition [27, 39].

Our work uses a hybrid between the local and global descriptor approach by detecting lines at instance-level and describing clusters of lines for place recognition. We propose a novel line detection method in 3D, based on LSD, that uses depth information to obtain the 3D lines and a viewpoint-invariant image of the line environments. We use instance-level line clusters as features and a learned visual-geometric descriptor to recognize places. No previous work demonstrated an approach that uses instance-level descriptors with structural information while taking advantage of the robustness of lines as visual features.

## 3. Method

In our approach, we first detect line segments in an RGB camera image using the LSD line detector [19]. Next, these

lines are reprojected to 3D using the depth map. We capture additional geometric information and a virtual image of the environment around each line. The line geometries and virtual images are then processed by a line clustering neural network, which predicts a cluster label for each line. Each group of lines with equal cluster label is then processed by a second neural network for cluster description to obtain an embedding for each line cluster. Both networks are based on attention mechanisms. The cluster description embedding can be compared to cluster embeddings from other frames to recognize clusters and scenes. The complete pipeline is shown in Figure 2.

### 3.1. Extraction of line information

In this section, we explain the inputs to the neural networks for clustering and cluster description and how they are generated from RGB-D images.

#### 3.1.1 Line detection and reprojection

Line segments are extracted from gray-scaled RGB images using the LSD line detector [19], which is robust and fast. Co-linear line segments with close extremities are fused. Lines below a certain pixel length are rejected. Point clouds of the line neighborhood are obtained by pooling the depth map with rectangular regions on either side of the line. A plane is fitted on each of the neighborhood point clouds using RANSAC. If the planes are co-planar (texture line) or only one plane exists (*e.g.* due to discontinuity), the line extremities are projected onto the corresponding plane. If the distance between the two plane centers along either of the surface normals is over a certain threshold (0.3 m), the line is projected onto the plane closer to the camera (discontinuity). Otherwise, the extremities are projected onto the intersection line of the two planes (edge).

#### 3.1.2 Geometric information of line segment

After extraction, information about the geometry of the line segments is recorded. It consists of the 3D start and end point  $p_{s/e}$ , and the normals  $n_{l/r}$  of the previously described neighborhood planes (Figure 3a). These planes give further structural information. If one plane does not exist, the corresponding normal is zero-padded. The line segment length  $l$  is precomputed and appended to the information. Notice that the line length is already encoded in the extremities of the line, but it is added explicitly to help the neural networks to take length into account. Lastly, two boolean values  $o_{s/e}$  are stored indicating whether the line extremities are occluded by an object in front of it or if they are on the edge of the view plane. The idea is that in case of occlusion, the actual position of extremities and length of the line is unknown and should thus be handled differently by the network. All

geometrical information is stored in a 15-dimensional vector

$$x_{i,geom} = [p_s, p_e, n_l, n_r, l, o_s, o_e,]^T_i \in \left( \begin{array}{c} \mathbb{R}^{13 \times 1} \\ \mathbb{B}^{2 \times 1} \end{array} \right). \quad (1)$$

#### 3.1.3 Virtual images of line environment

To gather additional appearance information from the available RGB data, a viewpoint invariant virtual image is taken from the environment of the line. The point cloud in the vicinity of the line is orthogonal-projected onto a virtual camera plane. To make the perspective of this virtual camera invariant to the viewing angle, the camera plane is set at a fixed distance of 0.5 m, parallel to the line and perpendicular to the mean of the two neighborhood plane normals. The camera has a maximum viewing distance of 1 m and the field of view corresponds to 1.5 times of the line length in width, and one time in height (Figure 3a). The image is then discretized with a variable resolution to prevent aliasing. Despite variable resolution, these images suffer from scattered empty pixels. To deal with these holes, a dilated mask of non-black pixels is inpainted using the Navier-Stokes algorithm [7] (Figure 3b). Finally, it is linearly resized to a resolution of  $96 \times 64$  and stored in the appearance information vector

$$x_{i,visual} \in \mathbb{R}^{96 \times 64 \times 3}. \quad (2)$$

### 3.2. Neural network for line clustering

The task of the clustering network is to predict a cluster label for each line. The neural network needs to process each line, while simultaneously gathering information of the surroundings of the line, similar to what CNNs do with pixels. The idea is that lines need to “know” their neighborhood lines to determine the correct correspondences. For example, the lines that make up a door frame are shaped like a door, and should therefore be assigned the same label. However, due to the unsortable nature of lines and the variable number of lines in each frame, CNNs or Recurrent Neural Networks (RNNs) cannot be used without enforcing some kind of sorting.

As a solution, scaled residual multi-head attention modules are used in the *Transformer* [51]. It has been deployed successfully for natural language processing (NLP) tasks. The attention modules enable each word to “communicate” with other words in the sentence to understand the context. In our work, these attention modules enable each line to gather information about its surrounding lines and the scene.

The inputs of the network are the geometric and visual information of each line, and the output is a probability distribution, where each bin corresponds to the probability of a line belonging to a certain cluster. The predicted cluster label is then determined through `argmax`. In the following, we describe in detail how the network is structured and

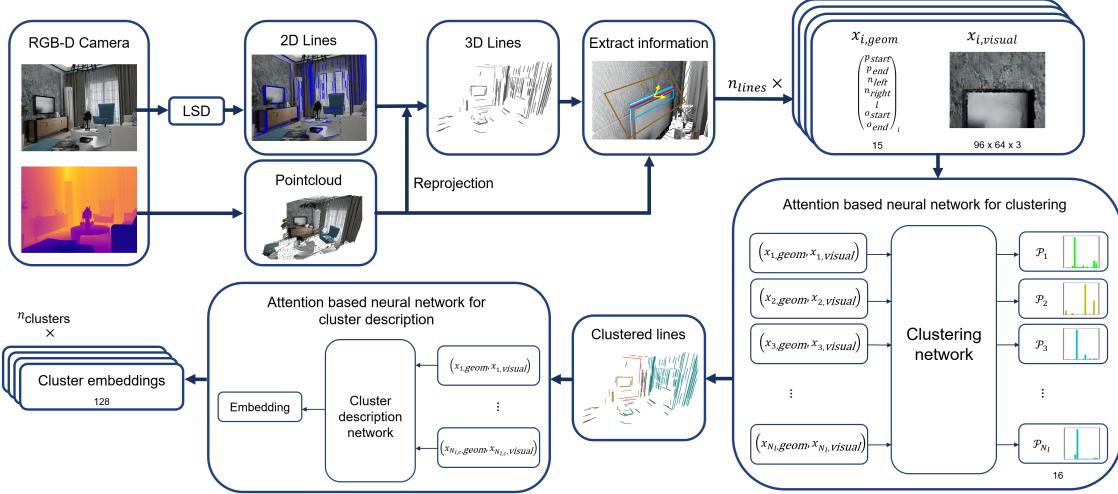


Figure 2: Summary of the proposed place recognition approach using line clusters as visual features.

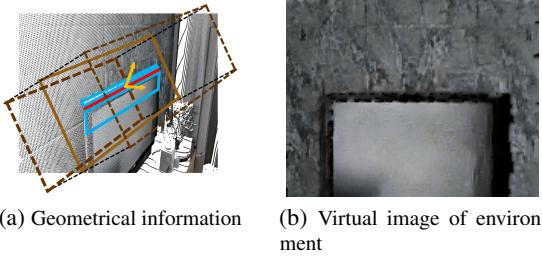


Figure 3: The geometrical information of a line together with a virtual camera image are used as the input to the neural network for clustering. The geometrical information (a) consists of the normals (yellow) of the adjacent planes (blue) of the line (red), the line length and whether the line ends are visible. The brown box indicates the view frustum of the virtual camera, resulting in the virtual image in (b).

trained, and how it achieves good clustering performance on indoor scenes.

### 3.2.1 Network architecture

The inputs of the network are the geometry and virtual image of each line  $\{(x_{i,geom}, x_{i,visual})\}_{i=1}^{N_l}$ , as described in Section 3.1.  $N_l \leq N_{l,max}$  is the total number of lines in a frame.

The geometric and visual data is preprocessed by the line embedding network. The geometric data is expanded with a linear layer and Leaky ReLu activation to form the  $d_{geom} = 472$  dimensional geometric encoding. The virtual image is passed through the first 7 layers of the VGG-16 CNN [43],  $4 \times 4$  max-pooled and flattened. A hidden layer with  $d_h = 2048$  neurons is added to output a visual encoding with a dimension of  $d_{visual} = 128$ . The geometrical and visual encodings are concatenated to form line embeddings of dimension  $d_{model} = 600$ . The line embedding networks run in parallel for every line and share the same weights.

Empty inputs are masked.

The main body of the network is composed of  $N_{layers} = 7$  identical layers adapted from the encoder modules from *Transformer* [51]. Each layer is made up of two sub-layers, the scaled multi-head self-attention layer and the position-wise fully connected (FC) feed-forward layer. Each sub-layer is bypassed with residual connections. Instead of normalization, Leaky ReLu activation with  $\alpha = 0.3$  is used as non-linearity. The residual output of each sub-layer is thus the Leaky ReLu activation of the sum of the input and the output of the sub-layer. In the attention sub-layer, four of the dot-product attention heads are replaced with additive attention heads, to handle the very diverse nature of the inputs. This results in  $h_{add} = 4$  additive self-attention heads and  $h_{dot} = 4$  dot-product self-attention heads per layer. The heads have a query/key dimensionality of  $d_q = d_k = 150$ . The key and value tensors are identical. The inner layer of the feed-forward network has dimensionality  $d_{ff} = 2400$  and Leaky ReLu activation with  $\alpha = 0.3$ . The dimension of all sub-layer outputs is  $d_{model}$  to facilitate the residual connections. Finally, a position-wise linear layer with a softmax activation outputs a probability distribution of dimension  $d_{label} = 16$ . Batch normalization is applied to all layer outputs to accelerate training. A graphical representation of the clustering network can be found in Figure 4.

### 3.2.2 Learning objective

The learning objective used in this work is adapted from [22], which proposed a method for pixel-wise instance segmentation. Detailed motivation of the following losses can be found in their paper.

Let us denote the output of the clustering network as the probability distribution  $\mathcal{P}_i = f(l_i) = [t_{i,bg}, t_{i,1}..t_{i,N_c}]$  where  $N_c = d_{label} - 1$  indicates the number of instance clusters detectable in each frame.  $t_{i,bg}$  is the probability of

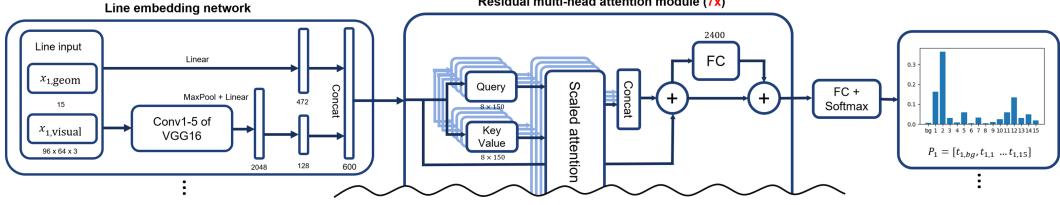


Figure 4: Part of the network architecture of the clustering network. The line embedding networks run in parallel for all lines.

line  $l_i$  to belong to the background cluster, and  $t_{i,k}$  is the probability of each line to belong to instance cluster  $k$ .

The task of the network is to assign a correct cluster label to each line. Lines of the same cluster should be assigned the same label, and lines of different clusters should be assigned different labels. We treat all background lines (*e.g.* floor, ceiling, wall lines) specially, as they tend to be spread out across the room, and are not packed together like typical instances (*e.g.* table, chair). The background label is absolute, and we reserve the first label  $t_{i,0}$  for those and apply a binary cross-entropy loss for background classification over all lines

$$\mathcal{L}_{bg} = \sum_i^{N_l} (I_i^{bg} \log t_{i,bg} + (1 - I_i^{bg}) \log (\sum_{k=1}^{N_c} t_{i,k})). \quad (3)$$

Cluster labels of instances, however, are not absolute, since they only contain relative information. Switching labels of two clusters results in equivalent clustering. The label assignment is not unique. We define the relationship between any two non-background lines  $l_i$  and  $l_j$  as

$$R(l_i, l_j) = \begin{cases} 1 & \text{if } l_i, l_j \text{ belong to the same instance,} \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

We apply a KL-divergence based loss, which prefers similar probability distributions for lines of same labels, and dissimilar distributions for lines of different labels. For line pairs of the same label we apply a cost given by

$$\mathcal{L}(l_i, l_j)^+ = \mathcal{D}_{KL}(\mathcal{P}_i^* || \mathcal{P}_j) + \mathcal{D}_{KL}(\mathcal{P}_j^* || \mathcal{P}_i),$$

where  $\mathcal{D}_{KL}(\mathcal{P}_i^* || \mathcal{P}_j) = \sum_{k=1}^{N_c} t_{i,k} \log(\frac{t_{i,k}}{t_{j,k}}).$  (5)

For line pairs of different label we apply the hinge loss

$$\mathcal{L}(l_i, l_j)^- = L_h(\mathcal{D}_{KL}(\mathcal{P}_i^* || \mathcal{P}_j), \sigma) + L_h(\mathcal{D}_{KL}(\mathcal{P}_j^* || \mathcal{P}_i), \sigma),$$

where  $L_h(e, \sigma) = \max(0, \sigma - e).$  (6)

We use  $\sigma = 2$  as margin. The contrastive loss is a combination of the above losses

$$\mathcal{L}(l_i, l_j) = R(l_i, l_j) \mathcal{L}(l_i, l_j)^+ + (1 - R(l_i, l_j)) \mathcal{L}(l_i, l_j)^-. \quad (7)$$

Let  $T = \{(l_i, l_j)\}_{\forall i,j}$  contain all non-background line pairs, then the full pairwise loss is

$$\mathcal{L}_{pair} = \sum_{(l_i, l_j) \in T} \mathcal{L}(l_i, l_j). \quad (8)$$

And finally we add the background and pairwise loss to formulate the full loss function for clustering

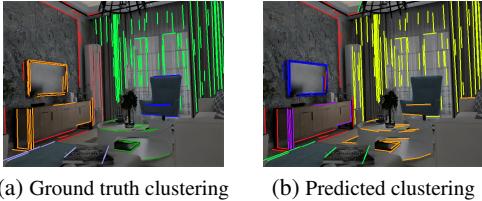
$$\mathcal{L}_{clustering} = \mathcal{L}_{pair} + \mathcal{L}_{bg}. \quad (9)$$

### 3.2.3 Training

The network is trained on the synthetic InteriorNet dataset [28]. It consists of more than 10k scenes with 20 photo-realistic RGB-D frames each from random viewpoints. The camera sensor is modeled as a pinhole camera with perfect depth data. Since it is a synthetic dataset, it often reuses the same object models. To prevent overfitting to these object models, a subset of 3600 scenes is selected for training, and a subset of 400 scenes is selected for validation. The dataset contains ground truth semantic and instance label masks. The ground truth labelling of the lines is determined by majority voting of the pixels in the line neighborhood.

The VGG layers of the line embedding network are initialized with ImageNet pretrained weights [40]. The visual encoding network is pretrained for 20 epochs using only visual data. During pretraining, the last VGG layers are gradually unfrozen until only the first two layers are left with the original ImageNet weights. The weights of the visual encoding network are again frozen during training of the full network. The full network is trained using the Adam optimizer [25] for 40 epochs, with an initial learning rate of  $5e^{-5}$  and a gradual decay. To limit memory usage during training, the maximum number of lines per frame is set to  $N_{l,max} = 160$ . During evaluation, the maximum number of lines per frame is set to  $N_{l,max} = 220$ . If a frame contains more than  $N_{l,max}$  lines, lines are sampled at random. Data augmentation is applied by rigidly rotating the frames and blacking out images at random.

Training took 6 days on an Nvidia RTX 2070 Super GPU. The average inference time for 220 lines is 0.1 s with a memory consumption of 3.5 GB.



(a) Ground truth clustering

(b) Predicted clustering

Figure 5: Comparison between Ground Truth Clustering (GTC) and our prediction. Clusters with less than 4 lines are not shown. Different colors represent different cluster labels. Red lines are background lines. Note that absolute cluster colors are arbitrary without meaningful correspondences from (a) to (b).

### 3.2.4 Clustering performance

Clustering performance is measured using the Normalized Mutual Information (NMI) metric [53]. Background lines are treated as a separate cluster. It is compared to the conventional agglomerative clustering method [13], where the minimum Euclidean distance between line segments is used as distance metric. The cluster count is determined using silhouette score minimization. To show that the attention layers indeed work, a vanilla FC network (with the same architecture, with the exception of removed attention layers) is also trained and tested. The average NMI over all frames of our validation subset of the InteriorNet dataset is determined for each method. Agglomerative clustering achieves an average NMI of 0.45 while the vanilla FC network improves on this with an average NMI of 0.56. Using our clustering approach, we outperform both methods by achieving an average NMI of 0.72 out of a perfect score of 1. The agglomerative clustering approach mostly struggles to find the correct number of clusters, and fails to differentiate adjacent clusters. Qualitative results of our method can be seen in Figure 5.

## 3.3 Neural network for cluster description

Next, we explain how we describe line clusters with a descriptor embedding. Line clusters are the groups of lines with the same predicted label. The maximum number of line clusters in each frame is thus  $d_{label}$ . All groups of lines are sequentially fed into the network that uses the same inputs as the network for clustering. The network output is a descriptor embedding in  $d_{descriptor} = 128$  dimensional euclidean space.

### 3.3.1 Network architecture

The descriptor network uses the same line embedding network structure and the same residual multi-head attention layers as the clustering network. The dimensionality of the visual encoding is the same. Because the clusters contain less lines than frames and thus less structural information, the dimensionality of the geometric encoding is reduced to  $d_{geom} = 384$ , making the line embedding dimension

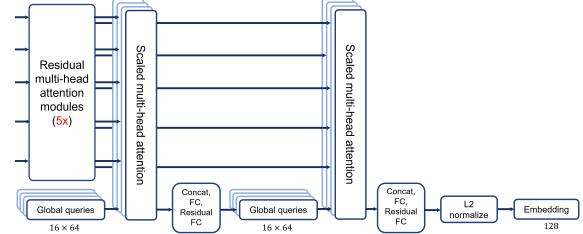


Figure 6: The tail part of the descriptor network consists of two scaled global multi-head attention layers gathering information over all lines using learned query vectors. This information is then concentrated into a normalized 128-dimensional embedding.

$d_{model} = 512$ . The dimensionality of the heads are reduced to  $d_k = d_v = 64$ . The hidden layer of the residual feedforward modules has a dimension  $d_{ff} = 2048$ . Furthermore, only  $N_{layers} = 5$  attention modules are used. At the output of the last attention module, two global attention modules are added (Figure 6). They work in the same way as the previously shown self-attention modules. Each line produces key/value vectors, and learned global query vectors “summarize” these values. The outputs of the attention modules are then concatenated. A linear layer with  $d_{global} = 1024$  activations is added and the output is fed through a residual FC layer. Linear layers then produce the global query vectors of the second global attention module. The outputs of this module are again concatenated and fed through a residual FC network. A final linear layer and L2 normalization is added to produce the descriptor embedding that lies on a 128-dimensional hyper sphere. The hidden layers of each residual FC layer have a dimension of  $d_{fc} = 4096$ . Both global attention modules have  $h_{dot} = 8$  and  $h_{add} = 8$  heads with a dimensionality of  $d_q = d_k = 64$ .

### 3.3.2 Triplet loss

We use the triplet loss as described in [57] for the object recognition learning task. We use triplets of clusters each consisting of an anchor, positive and negative cluster (Figure 7). The anchor and positive cluster are the same cluster viewed from another frame, and the negative cluster is a random different cluster. The objective is to increase the Euclidean distance between the anchor and negative cluster embeddings  $y_a$  and  $y_n$ , while decreasing the Euclidean distance between anchor and positive cluster embeddings  $y_a$  and  $y_p$ , respectively. The loss for each triplet in the training set is

$$\mathcal{L}_{triplet} = \max(0, ||y_a - y_p||_2 - ||y_a - y_n||_2 + m). \quad (10)$$

The margin  $m$  is a hyper parameter and set to 0.6.

### 3.3.3 Training

Weights of the visual encoding network are taken from the clustering network and frozen during training. We use the

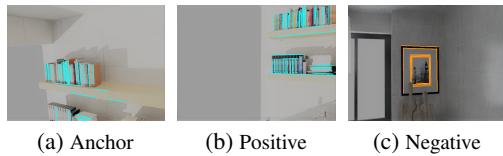


Figure 7: Example of a triplet used during training.

same subset of the InteriorNet dataset as described in Section 3.2.3. In the dataset, each instance has the same label across all 20 frames of each scene. The positive cluster pairs are extracted by selecting an instance with more than one occurrence across these 20 frames as the anchor cluster, and the same instance from another view as positive cluster. The negative cluster is an instance selected at random from another scene. With a probability of 0.5, the negative cluster is chosen to have the same semantic label as the anchor cluster. This is done to facilitate the differentiation of clusters of the same class. The network is trained using the Adam optimizer [25] for 50 epochs, with initial learning rate of 0.00001 and a gradual decay. The same data augmentation is applied as described in Section 3.2.3.

### 3.4. Place recognition

To perform place recognition, all RGB-D images from all scenes are passed through the proposed pipeline to obtain the line clusters and cluster embeddings for each frame. Clusters with less than 4 lines are removed to prevent matches based on insufficient information. Although omitted during the training of the cluster description network, the background is treated as any instance cluster during place recognition. The cluster embeddings of the map frames are assigned the corresponding scene index and stored in the map as a k-d tree for fast lookups. During query, the  $k_{nn}$  Nearest Neighbors (NNs) of each cluster embedding of the query frame are determined in the map. This results in  $n_{clusters} \times k_{nn}$  NNs for each frame, each assigned to a scene. The scene with most occurrences is selected and assigned to the query frame.

## 4. Evaluation

We evaluate the performance of the full place recognition pipeline on three datasets, and compare it to three baselines.

The performance of each method is measured by evaluating the ability to recognize scenes. The evaluation datasets consist of scenes with a number of frames each from different viewpoints. One frame is removed and the task is to assign this frame to the correct scene. This is repeated for every frame from every scene. The performance metric is the ratio of true positives (Accuracy), which is the ratio of correctly matched frames over the total number of frames.

We compare against a BoW approach using SIFT [29], and the learned SP [14] features. A vocabulary size of  $d_{sift} = 512$  on the NYU and DIML datasets, and  $d_{sift} = 800$  for the InteriorNet dataset is chosen for the SIFT + BoW

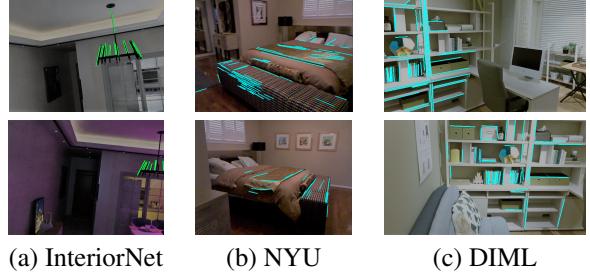


Figure 8: Examples of frames where the SIFT + BoW approach failed. Our method could recognize the scene by recognizing the objects, even under large viewpoint and lighting changes. The matched line clusters are indicated with colored lines.

approach. For the SP + BoW method, we use the network weights from the original paper with a vocabulary size of  $d_{sp} = 1024$ . The vocabulary is trained on the corresponding evaluation dataset. In addition to this, we compare to the global descriptor NetVLAD [10] along with the best performing model and weights (VGG16 + whitening, trained on Pitts30k). The NN frame is determined using the Euclidean distance between the vocabulary histograms. The query frame is then assigned to the scene of the NN frame.

### 4.1. InteriorNet dataset

The InteriorNet subset used for evaluation is the validation set used during training (Section 3.2.3). It contains 400 scenes, consisting of 20 frames captured in a single room. However, some of these scenes are connected, because some rooms are located on the same floor. We treat these rooms as the same scene during evaluation, resulting in 100 separate scenes. Large viewpoint changes are present because of these connected rooms. The number of NNs chosen for our approach is  $k_{nn} = 8$ . Since ground truth instance masks are available, our pipeline is also tested using GTC. That is, the neural network for clustering is replaced with ground truth cluster labels. Our approach outperforms the SIFT + BoW baseline by 52% and the more recent SP + BoW by 16%, as depicted in Table 1. The indoor environments of the InteriorNet dataset are very clean and consistent, thus local features are very repetitive (corners of different doors, drawers, tables, etc. appear the same), making the advantage of using high-level structural features apparent. The predicted line clusters are not as consistent between frames as the ground truth clusters, therefore the performance with GTC is improved. However, the strong performance of our full pipeline shows that the descriptor is robust to missing and wrongly assigned lines.

The InteriorNet dataset contains randomly illuminated versions of all frames. We tested the methods on illumination invariance by replacing the query frames with randomly illuminated frames (INet + RI), i.e. the map frames are regularly illuminated and the query frames are randomly illuminated.

For this evaluation, we retrained our descriptor and cluster model and included frames containing random lighting during training. Our pipeline performs better than the methods using local features (SIFT/SP + BoW), demonstrating the advantage of using line clusters under changing lighting.

NetVLAD significantly outperforms all other methods in all experiments. It benefits from the unique style of each scene designed by interior architects and can thus match frames even with no viewpoint overlap. Also, the dimensionality of the NetVLAD descriptor (4096) is significantly larger than ours (approx. 800 per frame).

Method	Accuracy			
	InteriorNet	IINet + RI	DIML	NYU
SIFT + BoW	38%	21%	66%	30%
SP + BoW	50%	32%	73 %	26%
NetVLAD	93%	91%	94%	84%
Ours	58%	34%	65%	20%
Ours, GTC	64%	41%	n/a	n/a

Table 1: Evaluation on the synthetic InteriorNet dataset, the InteriorNet dataset with random illumination (IINet + RI), the DIML dataset and the NYU dataset. Our approach performs significantly better than the SIFT + BoW approach on the InteriorNet dataset. Using perfect GTC, even better performance could be achieved. Results are slightly inferior on the real datasets because of depth noise and clutter.

## 4.2. DIML dataset

The DIML dataset [9] consists of a large number of real indoor and outdoor scenes. We use only indoor scenes, and pick only scene types that appear in the InteriorNet dataset, such as bedrooms, living rooms and kitchens. Our selection consists of 30 scenes with 8 frames each. We set  $k_{nn} = 4$ . The scenes of the DIML dataset are captured using the KinectV2 RGB-D sensor. The depth data is reprojected into the RGB camera frame and post-processed to fill any depth holes resulting from reprojection or missing data. The noisy depth data causes slightly offset line geometry and noisy virtual camera images. However, our pipeline still achieves similar results to the SIFT + BoW approach shown in Table 1, despite being trained exclusively on a synthetic dataset with perfect depth data. Our approach could recognize 26 frames that could not be recognized by the SIFT approach (Figure 8). The accuracy on the DIML dataset is higher than on the InteriorNet dataset for all approaches, because the number of scenes and frames is smaller.

## 4.3. NYU dataset

The NYU Depth Dataset V2 [35] is also comprised of a variety of indoor scenes. The RGB-D images were obtained using the Microsoft Kinect V1 sensor, and missing depth

data is filled using a colorization scheme. We use the labelled subset, and again select only scenes of types that appear in the InteriorNet dataset and contain 4 or more frames. The resulting selection consists of 52 scenes and 290 frames. We set  $k_{nn} = 8$ . Because this dataset is captured using an older version of the Kinect sensor, the depth and RGB data is less accurate and more noisy. Besides, the scenes are significantly more cluttered than the DIML dataset, which our approach is not trained to handle. These properties of the NYU dataset are reflected in the results shown in Table 1, where our approach performs slightly inferior to the SIFT + BoW and SP + BoW approach. Nevertheless, there are 35 frames recognized by our approach, but not by the SIFT + BoW approach, an example is shown in Figure 8.

## 5. Limitations and Outlook

Our approach typically works better with large viewpoint changes and lighting changes compared to SIFT + BoW and in some cases better than SP + BoW. It also handles clean and tidy rooms better. However, clutter adds numerous lines that impact the performance of the clustering network negatively. Therefore, in scenes with a lot of clutter, local keypoint approaches work better, as there are more keypoints provided there is no significant viewpoint change. Some DIML scenes and especially the scenes of the NYU dataset are very cluttered, explaining the weaker performance of our method. Furthermore, all frames of a scene are captured inside a single room for both real datasets, making the viewpoint changes limited. While our approach clearly outperforms the local feature based baseline approaches on simulated data, the performance decreases on real data. The cause of this decrease can be explained by having sparser and more corrupt virtual images as real world depth data has many holes and noise. A large portion of this issue could be solved directly by using more recent sensors that provide denser and more accurate depth images, such as the Kinect Azure. In addition, sequential depth frames can be fused to generate more complete virtual images. Rendering techniques that are more robust to empty pixels, such as surface generation or voxel based rendering, can be used as well.

Our place recognition performance can also be improved by adding other types of visual features such as keypoints, curves and ellipses thanks to the flexibility of our network architecture. Finally, the matched line clusters can be used to perform 6-DoF pose estimation and matches can be improved through geometric consistency.

## 6. Conclusion

We introduced a novel learning-based approach that uses line clusters for place recognition. While outperforming conventional image-retrieval methods, we show comparable performance to learning-based local features descriptors.

Even though, NetVLAD performs significantly better in recognizing a scene, our approach has the potential to be used in a 6-DoF localization pipeline. We showed how attention mechanism based neural networks can be used to both cluster lines and describe line clusters in RGB-D frames. These networks have a huge potential to also be trained and used for other clustering and description tasks. Additionally, we demonstrated a novel technique to describe lines with virtual camera images. This method improves viewpoint invariance and can be interesting for future research.

## References

- [1] C. Akinlar and C. Topal. Edlines: Real-time line segment detection by edge drawing (ed). pages 2837–2840, 09 2011. [2](#)
- [2] A. Angeli, S. Doncieux, J. Meyer, and D. Filliat. Incremental vision-based topological slam. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031–1036, 2008. [2](#)
- [3] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic. Netvlad: CNN architecture for weakly supervised place recognition. *CoRR*, abs/1511.07247, 2015. [1, 2](#)
- [4] R. Arandjelovic and A. Zisserman. All about vlad. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013. [2](#)
- [5] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918, 2012. [1](#)
- [6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. [1](#)
- [7] M. Bertalmio, A. Bertozzi, and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. volume 1, pages I–355, 02 2001. [3](#)
- [8] C. Cadena, D. Galvez-López, J. D. Tardos, and J. Neira. Robust place recognition with stereo sequences. *IEEE Transactions on Robotics*, 28(4):871–885, 2012. [2](#)
- [9] J. Cho, D. Min, Y. Kim, and K. Sohn. A large RGB-D dataset for semi-supervised monocular depth estimation. *CoRR*, abs/1904.10230, 2019. [8](#)
- [10] T. Cieslewski, S. Choudhary, and D. Scaramuzza. Data-efficient decentralized visual SLAM. *CoRR*, abs/1710.05772, 2017. [7](#)
- [11] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *I. J. Robotic Res.*, 27:647–665, 06 2008. [1](#)
- [12] M. Cummins and P. Newman. Highly scalable appearance-only slam - fab-map 2.0. 06 2009. [2](#)
- [13] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 01 1977. [6](#)
- [14] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017. [2, 7](#)
- [15] R. Finman, T. Whelan, L. Paull, and J. J. Leonard. Physical words for place recognition in dense RGB-D maps. In *ICRA workshop on visual place recognition in changing environments*, 05/2014 2014. [2](#)
- [16] F. Furrer. *Online Incremental Object-Based Mapping for Mobile Manipulation*. PhD thesis, ETH Zurich, Zurich, 2020-05. [2](#)
- [17] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. Nieto. Incremental object database: Building 3d models from multiple partial observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6835–6842, 2018. [2](#)
- [18] S. Garg, N. Sünderhauf, and M. Milford. Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics. *CoRR*, abs/1804.05526, 2018. [1](#)
- [19] R. Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32:722–32, 04 2010. [2, 3](#)
- [20] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. [2](#)
- [21] K. Ho and P. Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74:261–286, 07 2007. [2](#)
- [22] Y. Hsu, Z. Xu, Z. Kira, and J. Huang. Learning to cluster for proposal-free instance segmentation. *CoRR*, abs/1803.06459, 2018. [4](#)
- [23] Jin Han Lee, G. Zhang, J. Lim, and I. H. Suh. Place recognition using straight lines for vision-based slam. In *2013 IEEE International Conference on Robotics and Automation*, pages 3799–3806, 2013. [2](#)
- [24] Junqiu Wang, Hongbin Zha, and R. Cipolla. Combining interest points and edges for content-based image retrieval. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–1256, 2005. [2](#)
- [25] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. [5, 7](#)
- [26] M. Lange, F. Schweinfurth, and A. Schilling. Dld: A deep learning based line descriptor for line feature matching. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5910–5915, 2019. [2](#)
- [27] J. H. Lee, S. Lee, G. Zhang, J. Lim, W. K. Chung, and I. H. Suh. Outdoor place recognition in urban environments using straight lines. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5550–5557, 2014. [2](#)
- [28] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *British Machine Vision Conference (BMVC)*, 2018. [5](#)
- [29] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. [1, 7](#)

- [30] S. Lowry, N. Sünderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, pages 1–19, 11 2015. [1](#) [2](#)
- [31] Y. Lu and D. Song. Robust rgbd odometry using point and line features. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3934–3942, 2015. [2](#)
- [32] Y. Lu and D. Song. Robustness to lighting variations: An rgbd indoor visual odometry using line segments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 688–694, 2015. [2](#)
- [33] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, pages 1–57, 2020. [2](#)
- [34] B. Micusik and H. Wildenauer. Descriptor free visual indoor localization with line segments. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3165–3173, 2015. [2](#)
- [35] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. [8](#)
- [36] A. Oliva and A. Torralba. Torralba, a.: Building the gist of a scene: The role of global image features in recognition. progress in brain research 155, 23-36. *Progress in brain research*, 155:23–36, 02 2006. [2](#)
- [37] R. Paul and P. Newman. Fab-map 3d: Topological mapping with spatial and visual appearance. In *2010 IEEE International Conference on Robotics and Automation*, pages 2649–2656, 2010. [2](#)
- [38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. [1](#)
- [39] F. Poggenhans, A. Hellmund, and C. Stiller. Application of line clustering algorithms for improving road feature detection. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2456–2461, 2016. [2](#)
- [40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. [5](#)
- [41] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. [2](#)
- [42] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. pages 357–360, 01 2007. [2](#)
- [43] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014. [4](#)
- [44] Sivic and Zisserman. Video google: a text retrieval approach to object matching in videos. In *ICCV 2003*, volume 9, pages 1470–1477 vol.2, 2003. [1](#) [2](#)
- [45] R. Spezialetti, S. Salti, L. Di Stefano, and F. Tombari. 3d local descriptors—from handcrafted to learned. *3D Imaging, Analysis and Applications*, pages 319–352, 2020. [2](#)
- [46] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461, 2016. [2](#)
- [47] N. Sünderhauf, S. Shirazi, A. Jacobson, E. Pepperell, F. Dayoub, B. Upcroft, and M. Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. 07 2015. [1](#) [2](#)
- [48] X. Tang, W. Fu, M. Jiang, G. Peng, Z. Wu, Y. Yue, and D. Wang. Place recognition using line-junction-lines in urban environments. pages 530–535, 11 2019. [2](#)
- [49] C. Toft, E. Stenborg, L. Hammarstrand, L. Brynte, M. Pollefeys, T. Sattler, and F. Kahl. Semantic match consistency for long-term visual localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 383–399, 2018. [2](#)
- [50] C. Valgren and A. Lilienthal. Sift, surf and seasons: Long-term outdoor localization using local features. 09 2007. [2](#)
- [51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. [3](#) [4](#)
- [52] B. Verhagen, R. Timofte, and L. Van Gool. Scale-invariant line descriptors for wide baseline matching. In *IEEE Winter Conference on Applications of Computer Vision*, pages 493–500. IEEE, 2014. [2](#)
- [53] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, Dec. 2010. [6](#)
- [54] L. Wang, U. Neumann, and S. You. Wide-baseline image matching using line signatures. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1311–1318. IEEE, 2009. [2](#)
- [55] Z. Wang, F. Wu, and Z. hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42:941–953, 05 2009. [2](#)
- [56] T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016. [2](#)
- [57] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. *CoRR*, abs/1502.05908, 2015. [6](#)
- [58] A. B. Yandex and V. Lempitsky. Aggregating local deep features for image retrieval. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1269–1277, 2015. [1](#)
- [59] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgbd reconstructions. In *CVPR*, 2017. [2](#)
- [60] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh. Building a 3-d line-based map using stereo slam. *IEEE Transactions on Robotics*, 31(6):1364–1377, 2015. [2](#)
- [61] L. Zhang and R. Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013. [2](#)