Efficient Large-Scale Semantic Visual Localization in 2D Maps

Tomas Vojir[1,2][0000−0001−7324−5883], Ignas Budvytis[1][0000−0002−4278−198X], and Roberto Cipolla[1][0000−0002−8999−2151]

[1] University of Cambridge, Cambridge, UK
[2] Czech Technical University in Prague, Prague, CZ

Abstract. With the emergence of autonomous navigation systems, image-based localization is one of the essential tasks to be tackled. However, most of the current algorithms struggle to scale to city-size environments mainly because of the need to collect large (semi-)annotated datasets for CNN training and create databases for test environment of images, key-point level features or image embeddings. This data acquisition is not only expensive and time-consuming but also may cause privacy concerns. In this work, we propose a novel framework for semantic visual localization in city-scale environments which alleviates the aforementioned problem by using freely available 2D maps such as OpenStreetMap. Our method does not require any images or image-map pairs for training or test environment database collection. Instead, a robust embedding is learned from a depth and building instance label information of a particular location in the 2D map. At test time, this embedding is extracted from a panoramic building instance label and depth images. It is then used to retrieve the closest match in the database.

We evaluate our localization framework on two large-scale datasets consisting of Cambridge and San Francisco cities with a total length of drivable roads spanning over 500 km and including approximately 110k unique locations. To the best of our knowledge, this is the first large-scale semantic localization method which works on par with approaches that require the availability of images at train time or for test environment database creation.

1 Introduction

In this work, we propose a novel framework for semantic visual localization in 2D maps. Our work is motivated by the need for easy-to-use, scalable localization methods. One of the biggest challenges in achieving this goal includes the need of collecting large semi-annotated datasets of images [1–5], point clouds [6, 7] or key-point level features [8] which is not only computationally expensive but also causes additional privacy concerns. In localization task, image datasets are collected for two reasons: (i) to learn image or key-point level features and embeddings [4, 5, 9, 3], and (ii) to collect a database of images, image-level embeddings or 2D features of the test environment [4, 5, 9, 3, 10, 11, 7]. In most deep

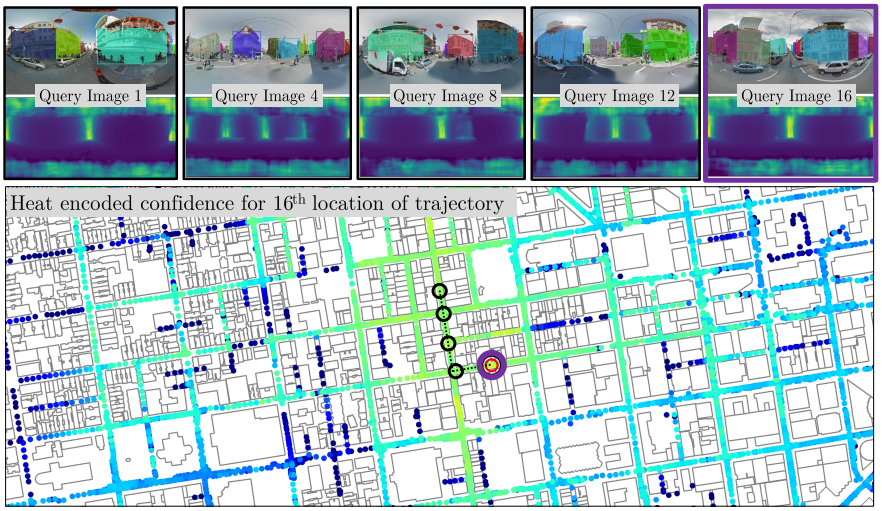


Fig. 1. This figure illustrates the typical results of our large scale semantic localization method. The top two rows show input query images with overlayed estimated building instances and estimated depth images for five locations of the query trajectory (marked by black circles). The bottom image shows the map locations similarity score for the last trajectory location encoded as a heat map (blue - low, red - high). The red circle marks the estimated most confident location for the 16^{th} query image and the ground truth location is marked in purple.

network approaches [10, 11, 7] or in conventional feature-based approaches [12, 13], images from the same or sometimes similar [3] environments are used. In contrast, our method does not require any images in order to train our embedding network or to build a database of test environments. Instead, it leverages the freely available 2D maps (e.g. OpenStreetMap [14]). Hence, our approach allows us to train a framework for localization on large cities in less than 24 hours effortlessly following the steps explained below.

Our method consists of four key steps. Firstly, a 2D map of a city of interest is downloaded from OpenStreetMap [14]. Secondly, a set of 2D map locations are sampled from the map. For each location, a semantic descriptor which encodes surrounding building presence as well depth profile is extracted. Thirdly, an embedding network is trained to learn the mapping of semantic descriptors to a low dimension space. The mapping ensures that descriptors of spatially close locations in the 2D map are similar in the embedding space and vice-versa. Finally, at test time, embeddings are calculated on semantic descriptors extracted from the building instance label [15, 16] and depth images [17] of the query image. They are then compared to the database of embeddings of the test environment locations to retrieve the most similar location.

The contributions of our work are as follows: (i) a novel semantic descriptor which encodes building instances and depth which can be efficiently extracted

from both 2D map as well as from an RGB image, and (ii) a novel efficient localization framework which can be trained fully from 2D map data without any annotated images for training or for building a database for the test environment.

We evaluate our method on 110k Google StreetView images from Cambridge and San Francisco cities. Our dataset covers more than 500km of the total driving length, of which query trajectories cover 200km. We significantly outperform binary semantic descriptor (BSD [18, 19]) on both single location retrieval and trajectory retrieval. A typical trajectory localization result obtained by our network is shown in Figure 1. We also outperform the most recent large scale localization approach [3], which requires the collection of images for training of their method, at the task of trajectory retrieval and demonstrate less sensitivity to the miss-match between training and test datasets. To the best of our knowledge, this is the first large scale semantic localization method which does not require images to train embedding network nor to build a representation of the test environment and works on par with localization techniques [3] which require the availability of images for training.

The rest of this work is divided as follows. Section 2 discusses relevant work in localization. Section 3 provides details of our proposed localization method. Sections 4 and 5 describe the experiment setup and corresponding results.

2 Related Work

In this section, we provide a description of related work for retrieval-based, end-to-end trained, and very large-scale localization methods.

Localization by image retrieval. Image retrieval localization methods [8, 20, 21, 1, 2, 4, 5, 9] work by finding the visually most similar image in a database of previously collected set of images and reporting the associated image pose of the best match. Images in the database are commonly represented by a collection of local features (e.g. SIFT [22]) which are often aggregated to a single feature vector per image (e.g. bag-of-visual-worlds [12], Fisher Vector [13]). To reduce the storage of the database, the compression of aggregated features is proposed [23] for efficient storage and fast matching of query images. To improve the localization pipeline, feature extraction, aggregation and compression stages can be replaced by end-to-end learned image representations [4, 5, 9] which yield better performance than the methods using hand-crafted features. The accuracy of the localization, assuming that correct closest database images can be retrieved, depends significantly on how densely the environment is sampled in order to build comprehensive database of the images. To refine the localization accuracy, several approaches have been proposed [6, 24, 25] that extend the retrieval-based localization by 3D information enabling precise 6-DoF camera pose estimation. These methods use either (i) global 3D reconstruction [26–29] from all database images [6, 24] and then establish 2D-3D matches to estimate the query image camera pose by PnP [30, 31] algorithms or (ii) local 3D reconstruction from retrieved images [25] followed by a resection of the query image into the local reconstruction. There are two common issues which are encountered when per-

forming large-scale localization: (i) the requirement of large storage space for the database, which is usually tackled by trying to compress the database without the loss in localization accuracy [32, 25] and (ii) the high self-similarity of feature descriptors and local geometry [33] of unrelated images in large-scale datasets, which hinders the accuracy of successful retrieval of closest images.

End-to-End localization. There are two general approaches in end-to-end deep learning based localization. The first set of approaches can be described as direct pose regression [34, 10, 35, 36] methods. These methods train an end-to-end network to regress camera pose for a given input image. The training data consist of pairs of images and corresponding camera poses, where the camera pose is usually represented by GPS coordinates (coarse localization [36]) or camera position and orientation (6-DoF accurate localization [34, 10, 35]). It was shown recently by [37, 7], that these direct pose regression approaches are more similar to image-based retrieval methods. The results for unseen test images are internally computed by interpolating poses of training images rather than by estimating accurate camera pose via the 3D structure of the scene. The training data size and the distribution of the training poses in the test environment are crucial for these methods. The second set of approaches attempt to regress 3D scene coordinates [11, 7] for each image pixel followed by absolute pose estimation algorithm (whether differentiable [38] or not [30, 31]) for accurate 6-DoF camera pose estimation. These methods generalize better for unseen test images than direct pose regression methods, but require structured training data (e.g. 3D mesh or point cloud) and are hard to scale to large scenes [38] because of training convergence difficulty or lack of training data. The scaling problem was partially addressed by [7] where the task of global scene coordinate regression was separated into two tasks of object recognition and local coordinate regression. This approach granted increased convergence speed and test accuracy. However, long training times and storage requirements of the 3D scene coordinates still prevent the scene coordinate regression works from being easily applied for city-size level environments.

Very large scale localization. There are very few approaches which tackle very large-scale localization problems on maps of size larger than $5km \times 5km$. Examples of such works include [39], who proposed a method for localization by matching a query trajectory estimated by visual odometry [40] to a GPS map of road segments. This method is complementary to ours rather than a direct competitor. Also, note, they use trajectory shapes to localize and need "unique" road topology to be successful, where our method can work in urban environment with Manhattan-like road topology, as demonstrated in San Francisco city (see Section5). Methods proposed in [18, 19] use a very small and efficient-to-compute hand-crafted 4-bit binary semantic descriptor (BSD), which encodes the presence of junctions and building gaps in panoramic images. While Abonce et al. [3] proposed utilising a twin CNN-based embedding architecture for image and map-tile respectively for fast and efficient retrieval. The main disadvantage of [3] compared to our method is that it requires GPS annotated images for training, whereas we need only the map data. The method [3] also relies on the

network to learn geometric and semantic information from images and map-tiles in a way suitable for this cross-modality matching, which is prone to overfitting. On the other hand, we explicitly provide structured semantic information during training (depth and building labels), which improves generalization (see Section 5). Moreover, our method is easily adaptable to a new city (downloading a 2D map and training) where [3] needs to collect a new set of training images. Our work is most similar to BSD [18, 19] as we do not use images at train time, which enables us to very quickly deploy our localization method in new large-scale environments. However, unlike [18, 19], our method outperforms image-based approache [3], which use images at train time, on several trajectory retrieval metrics.

3 Method

Our proposed semantic localization method consists of four key steps: (i) collecting the 2D map dataset, (ii) obtaining semantic descriptors for road locations in the 2D map, (iii) training of an embedding network for the aforementioned descriptors, and (iv) retrieving most similar individual locations and trajectories in the database for a given semantic descriptor extracted from the query image. These steps are explained in more detail below and in Figures 2 and 3.

Obtaining 2D map data. The first step of our localization framework involves obtaining the 2D map data for the training of the embedding network and for the creation of the embeddings database of the test environment. OpenStreetMaps [14] service is used to download a 2D map, which contains a set of 2D polygons or 2D lines representing objects such as buildings or road centerlines. In this work, only building outlines are used for localization, however, other objects such as trees, road signs or road markings could be used. A partial illustration of a downloaded 2D map of the Cambridge (UK) city is shown at the top of Figure 2. Note that the full map used in our experiments reported in Section 5 cover approximately a $5 \times 5\text{km}^2$ area, which contains approximately 22000 buildings.

Obtaining semantic descriptors. The second step of our method involves extracting a semantic descriptor of an arbitrary location. This descriptor encodes two types of information about the location: the instance boundaries between buildings and corresponding distances between the query location and the buildings surrounding it. It is designed in such a way that it could be extracted both from a 2D map and from depth and building instance label images. In more detail, the following procedure is used to extract the semantic descriptor from a 2D map. For a given location $\mathbf{x}_i$ on a 2D map M, a quantised set of viewing directions $\Theta = \{\frac{i2\pi}{B}; \forall i \in N, s.t. 0 \leq i \leq B-1\}$ are considered. For every viewing direction $\theta \in \Theta$, a ray is cast from the location of interest. For each ray the distance d_θ and corresponding building identity b_θ of the closest intersection is recorded. The vector $\mathbf{b} = [b_0, b_1, ...b_{B-1}]$ is transformed to vector $\mathbf{e} = [e_0, e_1, ...e_{B-1}]$ where $e_i = \exp\left(\frac{-0.5}{\sigma^2}\left[\min_{j \in \{l | b_l \neq b_{l+1}\}}(i-j \mod B)\right]^2\right)$, i.e.

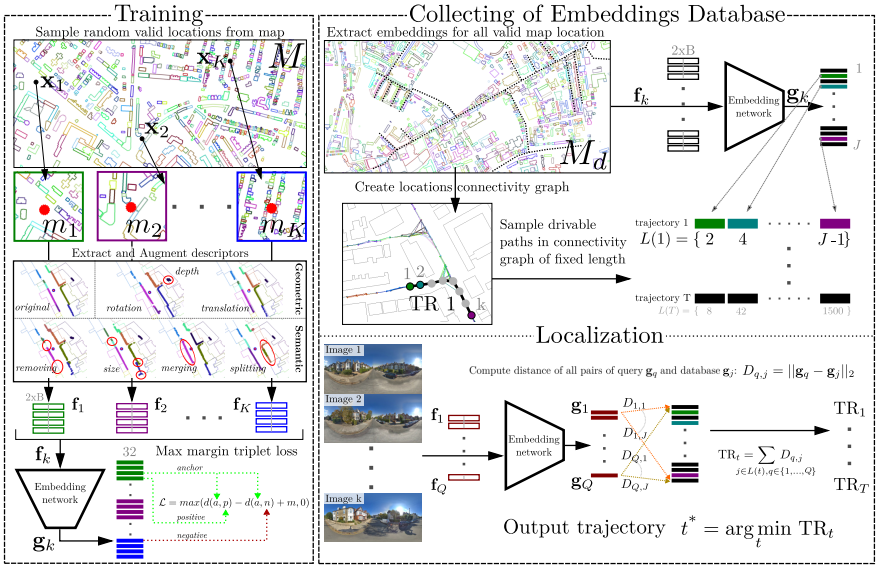


Fig. 2. This figure illustrates the key steps of our approach. During training, we randomly sample locations $\mathbf{x}_k$ from the 2D map M and extract the corresponding semantic descriptors $\mathbf{f}_k$ from visible local map m_k. Note that the extracted descriptors are semantically and geometrically augmented. The embedding network is then trained using max margin triplet loss, where positive samples come from augmentations and negative from different locations. To build a database for the test environment, semantic descriptors and their embeddings are extracted for J road locations (visualized as black dots in the map) of the 2D map M_d and possible drivable trajectories, $L(t), t \in \{1, ..., T\}$, are generated given the map locations connectivity graph. At localization time, the predicted labels and depth for each query image are used to extract a semantic descriptor $\mathbf{f}_q$ which is then passed through the learned embedding network. The individual embeddings $\mathbf{g}_q$ for query locations are compared with all database map embeddings $\mathbf{g}_j$ using Euclidean distance and the most similar trajectory, computed as the minimum sum location-wise distance $D_{q,j}$, is reported as the output.

each element encodes the distance to the closest building edge. This representation removes building identities and allows for a robust matching with noisy building labels extracted from query images during the localization procedure. An example of such descriptor is illustrated in Figure 3. The combined descriptor is obtained by simply concatenating $\mathbf{d} = [d_0, d_1, ...d_{B-1}]$ and $\mathbf{e}$ resulting in tensor $\mathbf{f}$ of shape $B \times 2$. At test time, building instance label and depth images are extracted using an off-the-shelf detection [15], semantic segmentation [16] as well as depth estimation [17] networks. Building instances with low Mask-RCNN [15] detection score (< 0.9) are filtered out. For each discretized angle θ, the most common building for that angle is obtained and transferred to the

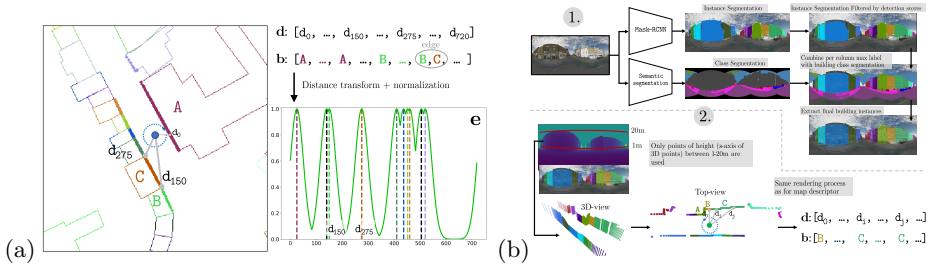
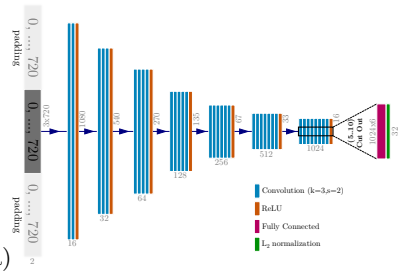


Fig. 3. The figure (a) illustrates the process of obtaining a semantic descriptor by casting rays uniformly around a given location in the map obtained from OpenStreetMap [14], and recording the distance of the first intersection $\mathbf{d}_\theta$ and building label $\mathbf{b}_\theta$ for each ray. The building labels are then converted to building edges and transformed to $\mathbf{e}$. The final descriptor is obtained by simply concatenating $\mathbf{d}$ and $\mathbf{e}$. The right figure (b) illustrates the process of extracting the semantic descriptor from the query image. In the first phase, the Mask-RCNN [15] is used to detect building instances. For each discretized column of the building instance segmentation image the most common building label is computed and used to label the corresponding columns of high resolution building class segmentation [16]. The second phase combines estimated depth in the height range of 1m-20m with the high-resolution instance segmentation to create a local map for which the same ray-casting procedure as for figure (a) is used to obtain $\mathbf{d}$ and $\mathbf{e}$ components of the final descriptor $\mathbf{f}$.

high resolution building segmentation[3] to obtain the final building instance segmentation masks. For each pixel with a valid building label its' corresponding 3D point is computed from the depth image [17], resulting in the same type of semantic descriptor as extracted from the 2D map. The 3D points outside of the height range of 1m-20m (assuming ground plane corresponds to the x-y plane of the equirectangular image) are filtered out as this helps to remove most false detections in the sky, sidewalks and street areas. The remaining 3D points are orthographically projected onto the ground plane by ignoring the z-axis. The same procedure as for obtaining a semantic descriptor from the 2D map is then used. The whole descriptor extraction process is illustrated in Figure 3. Note that by explicitly using semantic and geometric information, we can utilize heavy semantically and geometrically meaningful augmentation during the embedding network training, hence, making the descriptor embeddings more robust to changes captured by the augmentations. More implementation details are provided in Section 4.

Descriptor embedding. The semantic descriptors $\mathbf{f}$ are embedded into 32 dimension vectors $\mathbf{g}$ using a 1D convolutional neural network. The architecture is inspired by the VGG network [41], where the context for features is progressively enlarged to accommodate relations between faraway buildings. The exact architecture is described in Section 4. The CNN is trained using max-margin

[3] Note, we found that semantic segmentation networks obtain higher quality segmentations of building class labels.



(a)

(b)

City	Manhattan	Pittsburg	Cambridge	San Francisco
Area (km^2)	35	35	25	18
Number of Images	95k	60k	33k	80k
Driving Length (km)	-	-	230	285
Train or Test	Train		Test	

Fig. 4. The left figure shows the architecture of the embedding network. The table provides information about the datasets used in our experiments.

triplet loss [42, 43], which encourages the similarity and dissimilarity of semantic descriptors from the same and different locations respectively. All triplets mining strategy [44] is used to generate triplets for batch training. As a result of requiring only 2D maps for training the embedding network, there is a virtually unlimited amount of training data available. More details for network training and data augmentations are described in Sections 4.

Location retrieval. To retrieve the set of most similar locations from the database, the semantic descriptor $\mathbf{f}_q$ is extracted for all query images I_q, where $q \in \{1, ..., Q\}$, and corresponding embeddings $\mathbf{g}_q$ are obtained using the aforementioned embedding network. The Euclidean distance: $D_{q,j} = ||\mathbf{g}_q - \mathbf{g}_j||_2$ is measured between all query embeddings $\mathbf{g}_q$ and database embeddings $\mathbf{g}_j$. This procedure is illustrated on the right side of Figure 2. In order to perform retrieval of trajectories, the trajectory is represented as an ordered set $L(t)$ of database indexes for all locations in the trajectory t in the order of trajectory traversal. The trajectory score TR_t for a trajectory t is calculated as a sum of the location-wise scores: $TR_t = \sum_{j \in L(t), q \in \{1,...,Q\}} D_{q,j}$. The most similar trajectory (i.e. the minimum score) is returned as an output. We follow the protocol of [3] for candidate trajectory generation as explained in Section 4.

4 Experimental Setup

In this section, we describe the details of the datasets, training procedure and evaluation protocol employed.

Datasets. Since public large scale datasets covering cities larger than $5km \times 5km$ with dense equirectangular images were not available at the time of publication we have collected our own dataset consisting of four cities: Pittsburgh, Manhattan, Cambridge (UK) and San Francisco. For each city, we obtained a detailed 2D building outline map from OpenStreetMap and a sample of Google StreetView images. For example, in case of Manhattan, a total area of $35km^2$ was covered containing 55k buildings and 95k Google StreetView images. The corresponding details for other cities are provided in Figure 4(b). In our experiments, we use

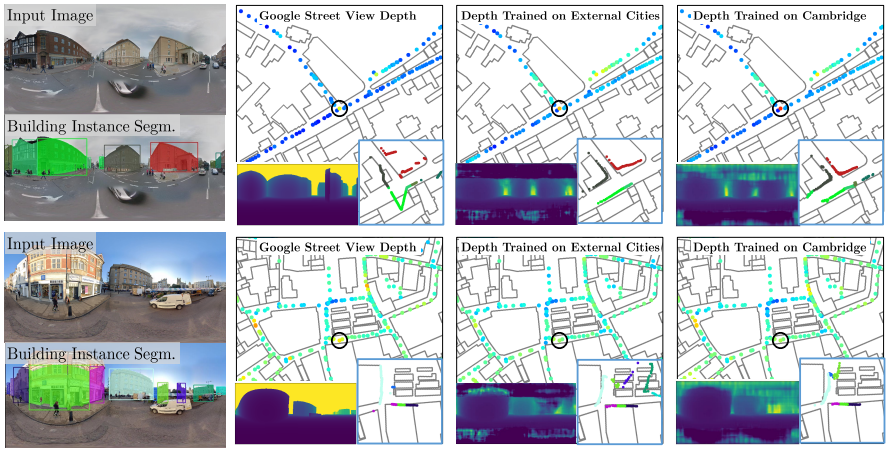


Fig. 5. This figure illustrates the qualitative performance of a single location retrieval of our method trained with Pittsburgh and Manhattan datasets. Three different columns on the right side correspond to different depths used at test time: (i) depth provided by Google StreetView (column 1), (ii) monocular depth estimation networks trained on external city images (column 2) and a similar network trained on Cambridge images (column 3). The quality of matches is encoded using a jet color scheme where best matches are encoded in red and worst matches in blue. Ground truth location is marked with a black circle. In the first row, due to the wrong depth of Google StreetView, the ground truth location produces a relatively weaker match than for depths obtained from monocular depth estimation networks. On the other hand, the second-row showcase where the images and map do not correspond and the depth estimated by the mono-depth approach hinders the localization. The results demonstrate the importance of depth quality for the localization.

OpenStreetMap data and images from Pittsburgh and Manhattan for training purposes (for competitors) and Cambridge as well as San Francisco for testing. When reporting accuracy results on Cambridge city we use San Francisco as validation data and similarly use Cambridge city as validation data for obtaining results on San Francisco. Note, however, since we do not require any images for training our embedding network it is also fair to use the 2D maps of the test environment for training.

Network architecture. The CNN architecture used for semantic descriptor embedding consists of seven sequentially stacked convolutional layers and a final dense layer with 32 output units with L_2 normalization of the 32 dimensional vector. For the convolutional part of the architecture the number of channels is doubled for every layer. The first convolutional block contains 16 channels and the final one contains 1024. All convolutional layers have ReLU non-linearities, zero padding, kernel size 3 and stride 2. The input descriptor is padded with itself from both sides to form a $3B \times 2$ input in order to avoid boundary effects in the network. Note that this padding is allowed by the circular property of our semantic descriptors. Only the valid features of the last convolutional layer

are passed to final fully connected layer. The illustration of the architecture is shown in Figure 4(a).

Training Data augmentation. For each descriptor of a given OpenStreetMap location, we perform two types of augmentation: geometric and semantic. Geometric augmentations are: (i) rotation of descriptor by $\pm 5^\circ$ (e.g. modeling missalignment of map and image), (ii) translation of descriptor location $\pm$5m (e.g. modeling imprecision of map location) and (iii) multiplicative depth noise per building $\pm 10\%$ and (iv) multiplicative depth noise per descriptor element $\pm 5\%$ of correct distances. The semantic augmentations are splitting buildings and merging neighboring buildings, making building shorter or larger, and removing buildings. Using these probabilities for semantic augmentation 0.5, 0.3, 0.3, 0.4 and 0.2 respectively. Examples of descriptor augmentations are shown in Figure 2 (left part).

Training Details. Adam optimizer with a learning rate of 0.0001 was used for training. For the building identity removal from descriptors a Gaussian with 0 mean and variance set to 5 was used. The variance was set empirically with slightly different values having a minimal effect. All the experiments reported on our method were trained in less than 24 hours on a single Titan X GPU.

Evaluation Protocol. Our localization framework is evaluated with respect to both single point and trajectory retrieval experiments. The quality of single point retrieval is examined by calculating the percentage of the successful retrieval of the query point for closest top 1%, 10% and 50% as shown in Figure 7(b). The accuracy of the trajectory retrieval is evaluated by the percentage of successfully retrieved trajectories. The retrieval is considered successful if the locations of final points of the query and retrieved trajectory are within $10m$ (a relaxed version of methodology used in [3]). For Cambridge and San Francisco cities we use 200 and 400 query trajectories correspondingly. They are matched against 200K and 500K alternative trajectories. Testing locations are taken to match GPS coordinates of StreetView images for compatibility with the evaluation protocol of [3, 18]. The alternative trajectories are generated by randomly traversing a graph made of testing locations as nodes and location connectivity as edges (see the illustration on the right side of Figure 2 and the supplementary material for implementation details). The generating traversal algorithm prevents trajectories from visiting the same location twice. Query trajectories are selected at random with a constraint that a medium number of buildings within trajectory should be larger than 3. See the visualization of query and database trajectories in the supplementary material.

5 Experiments

We evaluate our proposed localization method on two tasks: single point retrieval and trajectory retrieval, on different city combinations, and compare with competing methods [18, 19, 3]. For the first competing method, we simulated binary semantic descriptor BSD [18] estimation from images by replacing the learned classifiers of junctions and building gaps with a classification accuracy of 75% as

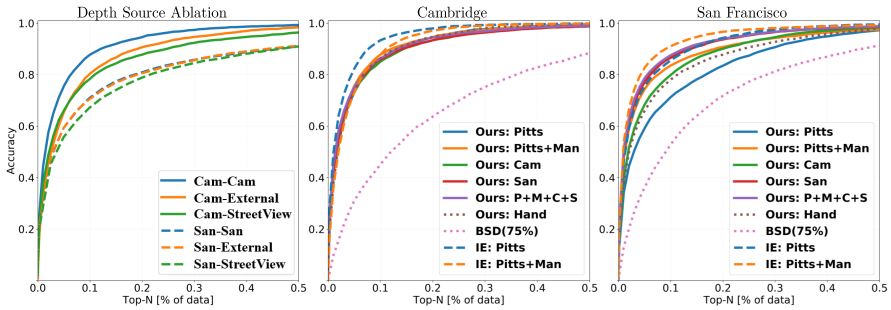


Fig. 6. The graph on the left shows the single point retrieval accuracy in top N% of locations. Our localization method is trained on Manhattan and Pittsburgh cities and tested using various depth predictions: (i) depth trained on the same city as tested on (blue), (ii) depth trained on external cities (orange) and (iii) depth accompanying the Google StreetView images (green). The depth trained on the same city achieves the highest accuracy highlighting the effect of good depth estimation on the retrieval performance. The middle and the right graphs show the same accuracy for various methods tested on Cambridge and San Francisco cities respectively. Our method trained on all four cities (P+M+C+S) achieved similar performance on both cities as best versions of IE for top-1% accuracy. However, unlike IE, our method performs consistently on both cities (note that the performance of IE versions differs significantly for Cambridge and San Francisco). For the full quantitative results please see Table 7.

in [3]. For the second comparison, we directly employed our descriptors (denoted as Hand-Crafted). Finally, we re-implemented the localization method of [3], denoted as Image Embedding (IE), following the guidelines provided in the publication and evaluated it in two settings - trained on Pittsburgh and jointly on Pittsburgh and Manhattan. We evaluated our method trained on five different sets of cities: (i) Cambridge, (ii) San Francisco, (iii) Pittsburgh, (iv) Pittsburgh and Manhattan as well as (v) all four cities. It can be seen in the Figures 8 and 7 that our network performs best when all four cities are used. Also, respectively a network trained on Cambridge data performs better on Cambridge query images than the one trained on the San Francisco city and the opposite. Since we do not use any images for training the embedding network, it is easy to leverage training data from 2D maps for test environments to boost performance, as demonstrated in these experiments.

5.1 Single Location Retrieval

In this experiment, we inspect the sensitivity of our method to errors in depth estimation and we compare our work to several state-of-the-art methods [19, 3, 18] in single query localization task.

Sensitivity to incorrect depth. We trained a semantic descriptor embedding network on a dataset consisting of the building outlines of Manhattan and Pittsburgh cities. At test time, depth images were obtained: (i) directly provided by

Method	Single Query Localization [Top x%]						Trajectory Localization [Top 1]								
	Cambridge		San Francisco		Average		Cambridge			San Francisco			Average		
	1%	10%	1%	10%	1%	10%	80m	160m	320m	80m	160m	320m	80m	160m	320m
BSD (75%)	9.6	45.1	13.1	52.9	11.4	49.0	6.9	27.2	70.0	1.3	6.2	21.3	4.1	16.7	45.7
Ours: Hand-Crafted	37.4	86.3	38.6	78.1	38.0	82.2	15.2	31.8	62.2	13.8	24.3	55.3	14.5	28.1	58.8
Ours: Pittsburgh	41.2	86.2	34.2	71.3	37.7	78.8	30.4	53.5	**88.0**	18.7	35.3	70.3	24.6	44.4	79.2
Ours: Pittsburgh + Manhattan	44.2	**87.8**	47.7	83.6	46.0	85.7	29.0	58.5	86.6	35.3	58.9	86.2	32.2	58.7	86.4
Ours: Cambridge	43.5	85.1	39.3	79.9	41.4	82.5	28.6	59.0	85.7	21.5	43.9	77.8	25.1	51.5	81.8
Ours: San Francisco	41.4	85.9	49.4	86.6	45.4	86.3	26.7	53.0	84.8	37.0	61.3	90.8	31.9	57.2	87.8
Ours: All (P+M+C+SF)	**45.1**	86.4	**51.4**	**87.6**	**48.3**	**87.0**	**30.9**	**62.7**	**90.8**	**42.8**	**72.7**	**93.8**	**36.9**	**67.7**	**92.3**
IE: Pittsburgh	**49.3**	**93.4**	46.5	85.6	**47.9**	**89.5**	**40.1**	**65.9**	85.7	34.4	65.8	90.3	**37.3**	**65.9**	**88.0**
IE: Pittsburgh + Manhattan	35.9	**87.8**	**53.9**	**91.1**	44.9	**89.5**	16.1	35.9	63.6	**42.6**	**71.4**	**92.0**	29.4	53.7	77.8

Fig. 7. This table shows the retrieval accuracy for top 1% and 10% best ranked single locations and the accuracy of the correct trajectory retrieval at different lengths of 80m, 160m and 320m for various methods. The "average" column shows a combined score over Cambridge and San Francisco cities. The IE stands for the Image Embedding method [3]. The best and second best results are in bold green and blue respectively. Our method performs favorably against IE variations and, in total, outperforms IE in average performance for trajectory retrieval. Note that our single method trained on all cities (highlighted in bold) achieves consistently high performance. In contrast, the performance of variations of IE method fluctuates significantly depending on the combination of train and test data.

the Google StreetView (Cam-StreetView and San-StreetView), (ii) from monocular depth estimation network trained on external cities (Cam-External and San-External) and (ii) from monocular depth estimation network trained directly on the images of Cambridge and San Francisco (Cam-Cam and San-San). The depth provided with the Google StreetView images was filtered and used to train the monocular depth networks as described above. Quantitative results are shown in Figure 6 and Figure 7. The best performance is obtained using a monocular depth network trained on the respective cities achieving 44.2% retrieval accuracy for top 1% predictions. Note that when testing using depth from a network trained on external cities the results have equally high accuracy on San Francisco city (48.1% at top-1%). As Google StreetView depth is provided using a planar approximation, some of the depth planes are missing or incorrectly placed, which significantly reduces the performance. Similarly, monocular depth estimation networks struggle to generalize in some cases and provide incorrect depth predictions. Figure 5 illustrates the qualitative results. For the rest of the experiments, we use the depth predictions obtained using the third method[4].

Comparison with alternative methods. Our network trained on all four cities achieved only a 4.2% and 2.5% lower top 1% accuracy on Cambridge and San Francisco respectively than IE [3]. On average, our method outperforms IE in the top 1% criterion, which shows that our model is more robust to the different test environments and less prone to overfitting. It is interesting to point out that single query localization performance correlates with the trajectory localization,

[4] Note that if LIDAR, stereo or other depth sensor data were available, they could be used instead.

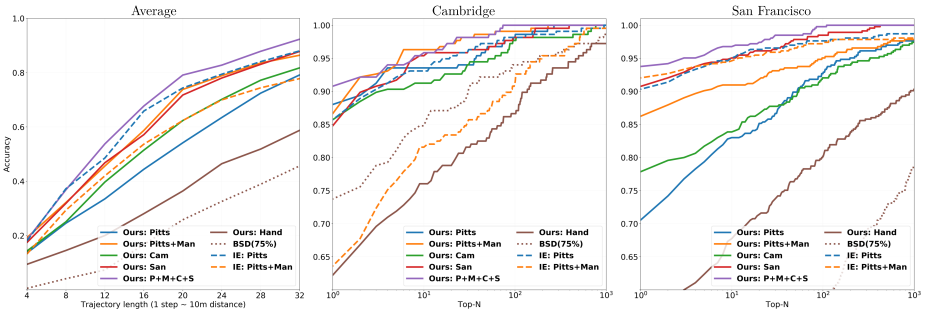


Fig. 8. The left graph shows the combined trajectory retrieval accuracy for Cambridge and San Francisco cities for varying trajectory length. Our method outperforms IE [3] in combined scores for almost all trajectory lengths. The middle and right graphs show the trajectory retrieval accuracy for 320m length trajectories on Cambridge and San Francisco, respectively, when considering top-N trajectories. Our method trained on all cities (P+M+C+S) is the top-performing with consistently high performance on both cities and converges to 100% retrieval accuracy when considering the top 100 most similar trajectories. Other methods need to consider ten times more trajectories.

but it is not the only factor. Our method performs better on average by 4.3% when longer driving scenario (> 200m) is considered as shown in Figure 7.

5.2 Trajectory prediction

For the final set of experiments, we evaluated the same networks as in Section 5.1 for the trajectory retrieval task. Figures 7 and 8 illustrate quantitative results. In particular, the left-most graph in Figure 8 shows that our network trained on all four cities achieves the highest combined (Cambridge + San Francisco) accuracy at all trajectory lengths larger than eight samples (≈ 80m). The right side of the Figure 8 show the trajectory retrieval accuracy for the trajectories of length 32 for Cambridge and San Francisco cities. It is important to note that the performance of the variants of Image Embedding [3] networks differ significantly when applied on Cambridge and San Francisco. The accuracy of the image embedding network trained on Pittsburgh and Manhattan cities decreases from 92.0% on San Francisco to 63.6% on Cambridge. This may be explained by the networks over-fitting to a particular dataset or type of city. All our networks, except for the one trained on Pittsburgh, show the biggest difference of 7.9% when evaluated on different cities. Two qualitative results demonstrating the strength and weaknesses of our and [3] methods are shown in Figure 9. In the left example (Cambridge), our method fails mainly because of severe occlusion by trees in most query images. Note that at the beginning of the trajectory, our method was more confident because of the cross-road and later it become less clear because of the occlusions. In contrast, the IE does not suffer from occlusions by trees and is able to localize correctly. However, note that the second best trajectory for IE is in the same location but the wrong direction and it

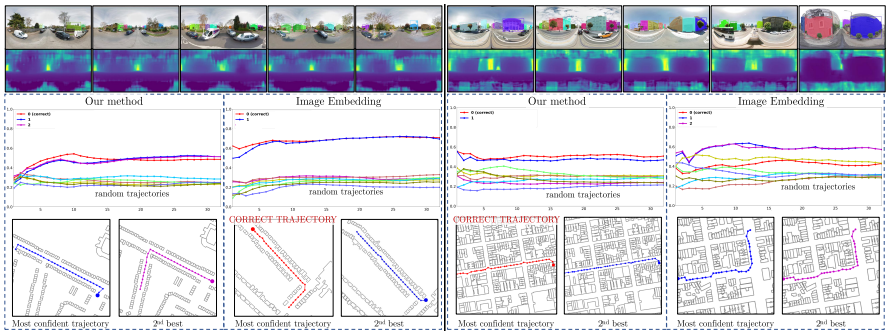


Fig. 9. This figure shows two examples of trajectory retrieval for Cambridge (left) and San Francisco (right). We show successful localization of our method (right) and a typical failure (left) and also compare it to IE [3]. The first two rows show input query images of the trajectory of length 32 with overlayed estimated building instances and depths. The plots in the middle show mean confidence of trajectories w.r.t. the number of query images. Each plot visualises scores for two most similar trajectories, the correct trajectory (red) and few random trajectories in order to illustrate confidence margins.

does not fully exploit cross-road structures (visible on the trajectory confidence graph). The second example on San Francisco (right part of the Figure) supports the previously mentioned IE issue that it does not learn the proper geometric structure and cannot take into account the width of the road.

6 Conclusions

We proposed a novel framework for semantic visual localization in 2D maps. Our work was motivated by the need for easy-to-use, scalable localization methods. Unlike most state-of-the-art large scale localization methods [6, 7, 3], our approach does not require any images at train time (either for training embedding or building database of the test environment). Hence, it enables us to train networks for localization on large cities in less than 24 hours effortlessly. We achieved this by making use of the following procedure. We first obtain a freely available 2D building outline maps from OpenStreetMap [14] and use them for training a semantic embedding network for a descriptor containing depth and building instances visible from particular locations on such a map. At test time, we extract a similar descriptor from the building instance segmentation and depth predictions of the actual query images and use it to find the closest match to various locations of the map in the embedding space. We evaluated our method on two cities of Cambridge and San Francisco. Our method significantly outperformed binary semantic descriptor (BSD [18, 19]) on both single location retrieval and trajectory retrieval. It also obtained higher combined retrieval accuracy than image-based localization method [3]. To the best of our knowledge, this is the first large scale semantic visualization method which works on par with localization techniques [3], which require images at train time.

References

1. Torii, A., Arandjelović, R., Sivic, J., Okutomi, M., Pajdla, T.: 24/7 place recognition by view synthesis. In: CVPR. (2015)
2. Budvytis, I., Sauer, P., Cipolla, R.: Semantic Localisation via Globally Unique Instance Segmentation. In: BMVC. (2018)
3. Abonce, O.S., Zhou, M., Calway, A.: You Are Here: Geolocation by Embedding Maps and Images. arXiv:1911.08797 (2019)
4. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. IEEE TPAMI (2018)
5. Schönberger, J., Pollefeys, M., Geiger, A., Sattler, T.: Semantic Visual Localization . In: CVPR. (2018)
6. Sattler, T., Havlena, M., Radenovic, F., Schindler, K., Pollefeys, M.: Hyperpoints and Fine Vocabularies for Large-Scale Location Recognition. In: ICCV. (2015)
7. Budvytis, I., Teichmann, M., Vojir, T., Cipolla, R.: Large Scale Joint Semantic Re-Localisation and Scene Understanding via Globally Unique Instance Coordinate Regression. BMVC (2019)
8. Arandjelovic, R., Zisserman, A.: DisLocation: Scalable Descriptor Distinctiveness for Location Recognition. In: ACCV. (2014)
9. Stenborg, E., Toft, C., Hammarstrand, L.: Long-Term Visual Localization Using Semantically Segmented Images. In: ICRA. (2018)
10. Kendall, A., Cipolla, R.: Geometric Loss Functions for Camera Pose Regression with Deep Learning. In: CVPR. (2017)
11. Brachmann, E., Rother, C.: Learning Less is More - 6D Camera Localization via 3D Surface Regression. CVPR (2017)
12. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
13. Perronnin, F., Liu, Y., Sánchez, J., Poirier, H.: Large-scale image retrieval with compressed Fisher vectors. In: CVPR. (2010)
14. Coast, S.: OpenStreetMap. (2004)
15. Massa, F., Girshick, R.: maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark (2018)
16. Zhu, Y., Sapra, K., Reda, F.A., Shih, K.J., Newsam, S., Tao, A., Catanzaro, B.: Improving Semantic Segmentation via Video Propagation and Label Relaxation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
17. Yakubovskiy, P.: Segmentation models. https://github.com/qubvel/segmentation_models (2019)
18. Panphattarasap, P., Calway, A.: Automated Map Reading: Image Based Localisation in 2-D Maps Using Binary Semantic Descriptors. In: IROS. (2018)
19. Yan, F., Vysotska, O., Stachniss, C.: Global Localization on OpenStreetMap Using 4-bit Semantic Descriptors. In: ECMR. (2019)
20. Chen, D.M., Baatz, G., Köser, K., Tsai, S.S., Vedantham, R., Pylvänäinen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., Grzeszczuk, R.: City-scale landmark identification on mobile devices. In: CVPR. (2011)
21. Schindler, G., Brown, M., Szeliski, R.: City-Scale Location Recognition. In: CVPR. (2007)
22. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV (2004)

23. Jegou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. IEEE TPAMI (2011)
24. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: ICCV. (2011)
25. Sattler, T., Torii, A., Sivic, J., Pollefeys, M., Taira, H., Okutomi, M., Pajdla, T.: Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In: CVPR. (2017)
26. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: ICCV. (2009)
27. Crandall, D., Owens, A., Snavely, N., Huttenlocher, D.: Discrete-continuous optimization for large-scale structure from motion. In: CVPR. (2011)
28. Schönberger, J.L., Radenović, F., Chum, O., Frahm, J.: From single image query to detailed 3D reconstruction. In: CVPR. (2015)
29. Schönberger, J.L., Frahm, J.: Structure-from-Motion Revisited. In: CVPR. (2016)
30. Kneip, L., Li, H., Seo, Y.: UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability. In: ECCV. (2014)
31. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: An Accurate O(N) Solution to the PnP Problem. IJCV (2009)
32. Camposeco, F., Cohen, A., Pollefeys, M., Sattler, T.: Hybrid Scene Compression for Visual Localization. In: CVPR. (2019)
33. Sattler, T., Havlena, M., Schindler, K., Pollefeys, M.: Large-Scale Location Recognition and the Geometric Burstiness Problem. In: CVPR. (2016)
34. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In: ICCV. (2015)
35. Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., Cremers, D.: Image-Based Localization Using LSTMs for Structured Feature Correlation. ICCV (2016)
36. Weyand, T., Kostrikov, I., Philbin, J.: PlaNet - Photo Geolocation with Convolutional Neural Networks. In: ECCV. (2016)
37. Sattler, T., Zhou, Q., Pollefeys, M., Leal-Taixé, L.: Understanding the Limitations of CNN-Based Absolute Camera Pose Regression. In: CVPR. (2019)
38. Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., Rother, C.: DSAC - Differentiable RANSAC for Camera Localization. In: CVPR. (2017)
39. Brubaker, M.A., Geiger, A., Urtasun, R.: Map-Based Probabilistic Visual Self-Localization. IEEE TPAMI (2016)
40. Nister, D., Naroditsky, O., Bergen, J.: Visual odometry. In: CVPR. (2004)
41. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: International Conference on Learning Representations. (2015)
42. Hermans*, A., Beyer*, L., Leibe, B.: In Defense of the Triplet Loss for Person Re-Identification. arXiv preprint arXiv:1703.07737 (2017)
43. Vo, N.N., Hays, J.: Localizing and orienting street views using overhead imagery. In Leibe, B., Matas, J., Sebe, N., Welling, M., eds.: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I. (2016)
44. Ding, S., Lin, L., Wang, G., Chao, H.: Deep feature learning with relative distance comparison for person re-identification. Pattern Recognition (2015) 2993 – 3003