Fine-Grained Segmentation Networks: Self-Supervised Segmentation for Improved Long-Term Visual Localization

Måns Larsson  Erik Stenborg  Carl Toft  Lars Hammarstrand  Torsten Sattler  Fredrik Kahl
Chalmers University of Technology

Abstract

Long-term visual localization is the problem of estimating the camera pose of a given query image in a scene whose appearance changes over time. It is an important problem in practice, for example, encountered in autonomous driving. In order to gain robustness to such changes, long-term localization approaches often use semantic segmentations as an invariant scene representation, as the semantic meaning of each scene part should not be affected by seasonal and other changes. However, these representations are typically not very discriminative due to the limited number of available classes. In this paper, we propose a new neural network, the Fine-Grained Segmentation Network (FGSN), that can be used to provide image segmentations with a larger number of labels and can be trained in a self-supervised fashion. In addition, we show how FGSNs can be trained to output consistent labels across seasonal changes. We demonstrate through extensive experiments that integrating the fine-grained segmentations produced by our FGSNs into existing localization algorithms leads to substantial improvements in localization performance.

1. Introduction

Visual localization is the problem of estimating the camera pose of a given image relative to a visual representation of a known scene. It is a classical problem in computer vision and solving the visual localization problem is one key to advanced computer vision applications such as self-driving cars and other autonomous robots, as well as Augmented / Mixed / Virtual Reality.

The scene representation used by localization algorithms is typically recovered from images depicting a given scene. The type of representation can vary from a set of images with associated camera poses [8, 75, 98], over 3D models constructed from Structure-from-Motion [77, 81], to weights encoded in convolutional neural networks (CNNs) [8, 10, 12, 13, 35, 36, 52] or random forests [11, 16, 79]. In practice, capturing a scene from all possible view-
self-supervised, data-driven approach to define fine-grained classes for image segmentation. More precisely, we use \( k \)-means clustering on pixel-level CNN features to define \( k \) classes. As shown in Fig. 1, this allows our approach, termed Fine-Grained Segmentation Networks (FGSNs), to create more fine-grained segmentations.

In detail, this paper makes the following contributions: 1) We present a novel type of segmentation network, the Fine-Grained Segmentation Network (FGSN), that outputs dense segmentation maps based on cluster indices. This removes the need for human-defined classes and allows us to define classes in a data-driven way through self-supervised learning. Using a 2D-2D correspondence dataset [42] for training, we ensure that our classes are stable under seasonal and viewpoint changes. The source code of our approach is publicly available \(^1\). 2) FGSNs allow us to create finer segmentations with more classes. We show that this has a positive impact on semantic visual localization algorithms and can lead to substantial improvements when used by existing localization approaches. 3) We perform detailed experiments to investigate the impact of the number of clusters on multiple visual localization algorithms. In addition, we compare two types of weight initializations, using networks pre-trained for semantic segmentation and image classification, respectively.

2. Related Work

The following reviews work related to our approach, most notably semantic segmentation and visual localization.

Semantic Segmentation. Semantic segmentation is the task of assigning a class label to each pixel in an input image. Modern approaches use fully convolutional networks [47], potentially pre-trained for classification [47], while incorporating higher level context [99], enlarging the receptive field [17, 19, 92], or fusing multi-scale features [18, 66]. Another line of work combines FCNs with probabilistic graphical models, e.g., in the form of a post-processing step [17] or as a differentiable component in an end-to-end trainable network [41, 46, 100].

CNNs for semantic segmentation are usually trained in a fully supervised fashion. However, obtaining a large amount of densely labeled images is very time-consuming and expensive [22, 55]. As a result, approaches based on weaker forms of annotations have been developed. Some examples of weak labels used to train FCNs are bounding boxes [23, 37, 57], image level tags [57, 59, 62, 82], points [9], or 2D-2D point matches [42]. In this paper, we show that the classes used for “semantic” visual localization do not need to carry semantic meaning. This allows us to directly learn a large set of classes for image segmentation from data in a self-supervised fashion. During training, we use 2D-2D point matches [42] to encourage consistency of the segmentations across seasonal changes and across different weather conditions.

(Semantic) Visual Localization. Traditionally, approaches for visual localization use a 3D scene model constructed from a set of database images via Structure-from-Motion [14, 20, 43–45, 73, 84, 95]. Associating each 3D model point with local image features such as SIFT [50], these approaches establish a set of 2D-3D correspondences between a query image and the model via descriptor matching. The resulting matches are then used for RANSAC-based camera pose estimation [26]. Machine learning-based approaches either replace the 2D-3D matching stage through scene coordinate regression [10, 12, 16, 52–54, 79], i.e., they regress the 3D point coordinate in each 2D-3D match, or directly regress the camera pose from an image [8, 13, 35, 36, 89]. The former type of methods achieves state-of-the-art localization accuracy in small-scale scenes [12, 16, 53], but do not seem to easily scale to larger scenes [12]. The latter type of methods have recently been shown to not perform consistently better than image retrieval methods [76], i.e., approaches that approximate the pose of the query image by the pose of the most similar database image [3, 38, 87]. As such, state-of-the-art methods for long-term visual localization at scale either rely on local features for matching [28, 71, 78, 83, 85, 86] or use image retrieval techniques [2–4, 63, 80, 87, 94].

One class of semantic visual localization approaches uses object detections as features [5, 6, 69]. In this paper, we focus on a second class of approaches based on semantic segmentations [4, 21, 28, 78, 80, 83, 85, 86, 94]. These methods use semantic image segmentations to obtain a scene representation that is invariant to appearance and (moderate) geometry changes. Due to the small number of classes typically available, the resulting representation is not very discriminative. Thus, semantic localization approaches use semantics as a second sensing modality next to 3D information [21, 78, 83, 85, 86]. In this paper, we show that the image segmentations used by such methods do not necessarily need to be semantic. Rather, we show that these approaches benefit from the more fine-grained segmentations with more classes produced by our FGSNs.

Domain Adaption. Semantic localization algorithms implicitly assume that semantic segmentations are robust to illumination, viewpoint, seasonal, and other changes. In practice, CNNs for semantic segmentation typically only perform well under varying conditions if these conditions are reflected in the training set. Yet, creating pixel-level annotations for large image sets is a time consuming task. Domain adaption approaches [27, 40, 48, 49, 68, 88, 101] thus consider the problem of applying algorithms trained on one

\(^1\)https://github.com/maunzzz/fine-grained-segmentation-networks
domain to new domains, where little to no labeled data is available. This makes training on synthetic datasets [65,67] to improve the performance on real images [33,70,102] possible. In addition, the performance of the network on images taken during different weather and lighting conditions can be improved [90,91]. In the context of (semantic) image segmentation, these approaches improve the robustness of the segmentations. However, they do not increase the number of available classes and are thus complimentary to our approach. We use a recently proposed correspondence dataset [42] for the same purpose, to ensure that our segmentations are robust to illumination and seasonal changes.

**Self-Supervised Learning.** Self-supervised learning approaches are a variant of unsupervised learning methods, where a model learns to predict a set of labels that can be automatically created from the input data. Several approaches train a CNN to perform a domain specific auxiliary task [25,56,61,97]. Some examples of tasks include predicting missing image parts [60], ego motion [1], and the rotation of an image [30]. To solve these auxiliary tasks, the CNNs need to learn meaningful visual features that can then also be used for the actual task at hand. In [15], Caron et al. train a CNN for the task of image-level classification using labels acquired by k-means clustering of image features. We extend this approach to training an image segmentation network. We also use the actual clusters, or labels, explicitly for visual localization. This in contrast to [15], where the clusters are just a means for learning features for tasks such as classification.

3. Fine-Grained Segmentation Networks

The Fine-Grained Segmentation Network (FGSN) has the same structure as a standard CNN used for semantic segmentation. Given an input image, it produces a dense segmentation map. However, instead of being trained on a set of manually created annotations, labels are created in a self-supervised manner. During training, at certain intervals, features are extracted from the images in the training set and clustered using k-means clustering. The cluster assignments, one at each pixel, are then used as supervision during training, i.e., as labels. In this way, we can change the number of classes that the FGSN outputs without having to create annotations with the new set of classes. The FGSN is trained to output the correct label for each pixel.

We also use a set of 2D-2D point correspondences [42] during training to ensure that the predictions are stable under seasonal changes and viewpoint variations. Each sample of the correspondence dataset contains two images of the same scene taken from different traversals and thus in different seasonal or weather conditions. One of the images in each pair is always from a reference traversal, captured during favourable weather conditions. A set of 2D-point correspondences between points in the images depicting the same 3D point is also available for each image pair. The network is encouraged to predict the same class for the two points in each correspondence to make the output robust to seasonal changes. Fig. 2 illustrates the training process. Note that creating the correspondence dataset is a significantly less laborious process than hand-labeling the same images with semantic labels, see details in [42].

**Label creation** For the creation of the labels we use the method developed by Caron et al. [15] based on k-means clustering. We, however, need to do some modifications to make it work well for dense output and training with 2D-2D correspondences. The main idea is to do k-means clustering on the output features of the CNN, then add a layer to the network and train using the cluster assignments as labels. After a fixed number of training iterations, the clustering is repeated and the final layer re-initialized.

For clustering we extract features from all images in the reference traversal of the correspondence dataset. This traversal contains images captured during favourable weather conditions, hence if we initialize the network with weights trained for semantic segmentation the features extracted will contain meaningful semantic information. For each image we get a dense map of image features, from which we randomly sample a set of features for clustering. Half of the features are extracted from pixel positions where we have 2D-2D correspondences and half are randomly sampled across the entire image. Given the set of extracted image features, clustering is done by solving

\[
\min_{C \in \mathbb{R}^{d \times m}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^m} \|d_n - Cy_n\|_2^2
\]

s. t. \(y_n^1m = 1\),

where \(d_n\) are feature vectors of length \(m\) sampled from the output feature maps produced by the CNN. Solving this problem provides a centroid matrix \(C^*\) and a set of optimal assignments \((y_n^*)\). To avoid trivial solutions with empty clusters we do a reassignment of the centroids of empty clusters. For each empty cluster centroid, a centroid of a non-empty cluster is randomly chosen. The centroid of the empty cluster is then set to the same value as this centroid with a small perturbation [15,34].

**Training Loss.** Our training loss consists of two parts, a correspondence part \(L_{corr}\) and a cluster classification part \(L_{class}\). The latter encourages the model to output the correct label for each pixel in the reference images of dataset. We use a standard cross-entropy loss with the labels as targets. The final \(L_{class}\) loss is an average of over all samples.

For \(L_{corr}\), we use the 2D-2D point correspondences. Denote the content of one sample from the correspondence dataset as \((I^r, I^f, x^r, x^f)\). Here \(I^r\) is an image from the
reference traversal, \( I^t \) is an image from the target traversal\(^2\), and \( x^r \) as well as \( x^t \) are the pixel positions of the matched points in the reference and target images, respectively.

The correspondence loss function \( L_{corr} \) is an average over all such samples

\[
L_{corr} = \frac{1}{M} \sum_{(r,t)} l_{CE}(I^r, I^t, x^r, x^t),
\]

where \( M \) is the number of samples and \( l_{CE} \) is the cluster correspondence cross-entropy loss. Let \( d_x \in \mathbb{R}^C \) denote the output feature vector of the network of length \( C \), the number of clusters, at pixel position \( x \). To calculate \( l_{CE} \) we begin by taking the cluster assignments, \( i.e. \) the labels, for the features in the reference image for all positions \( x^r \). By describing the label for a pixel at position \( x_i \) using the one-hot encoding vector \( e_{x_i} \), the loss can be written as

\[
l_{CE} = \frac{1}{N} \sum_{i=1}^{N} C_{x_i} \left( \log(d_{x_i}) + \log(d_{x_i}) \right),
\]

where \( \log(\cdot) \) is taken element-wise. The loss will encourage the pixels in the target image to have the same labels as the corresponding pixels in the reference image.

During training, we minimize \( L = L_{class} + L_{corr} \).

**Implementation Details.** During the training of the CNN, we minimize the loss \( L \) using stochastic gradient descent with momentum and weight decay. During all experiments the learning rate was set to \( 2.5 \cdot 10^{-5} \), while the momentum and weight decay were set to \( 0.9 \) and \( 10^{-4} \), respectively. We used the PSPNet [99] network structure with a Resnet101 [32] base. Due to GPU memory limitations we train with a batch size of one. The networks are trained for 60000 iterations and use the weights that obtained the lowest correspondence loss \( L_{corr} \) on the validation set. The training and evaluation are implemented in PyTorch [58].

After every 10000 iterations, a new set of image features are extracted from the reference images and a new set of cluster centroids and labels are calculated. The features, of initial dimension 512, are PCA-reduced to 256 dimensions, whitened and \( l_2 \)-normalized. The \( k \)-means clustering was done using the Faiss framework [34]. After clustering, the final layer of the network is randomly re-initialized using a normal distribution with mean 0 and standard deviation 0.01. The bias weights are all set to 0.

All evaluation and testing is done in patches of size 713 \( \times \) 713 pixels on the original image scale only. Patches are extracted from the image with a step size of 476 pixels in both directions. The network output is paired with an interpolation weight map that is 1 for the 236 \( \times \) 236 center pixels of the patch and drops off linearly to 0 at the edges. For each pixel the weighted mean, using the interpolation maps as weights, is used to produce the pixel’s class scores. The motivation behind the interpolation is that the network generally performs better at the center of the patches, since there is more information about the surroundings available.

**4. Semantic Visual Localization**

This paper was motivated by the hypotheses that being able to obtain more fine-grained image segmentations will have a positive impact on semantic visual localization approaches. To test this hypothesis, we integrate the segmentations obtained with our FGSNs into multiple semantic visual localization algorithms. In the following, we briefly review these algorithms. All of them assume that a 3D point cloud of the scene, where each 3D point is associated with a class or cluster label, is available. Since the point clouds are linked to images, the labels are obtained by projecting the segmentations of the images onto the point cloud.

**Simple Semantic Match Consistency (SSMC) [86].** The first approach is a simple-to-implement match consistency filter used as a baseline method in [86]. Given a set of 2D-3D matches between features in a query image and
3D points in a Structure-from-Motion (SfM) point cloud, SSMC uses semantics to filter out inconsistent matches. A match between a feature f and a 3D point p is considered inconsistent if the label of f obtained by segmenting the query image and the label of p are not identical. All consistent matches are used to estimate the camera pose by applying a P3P solver [31,39] inside a RANSAC [26] loop.

**Geometric-Semantic Match Consistency (GSMC) [86].** Assuming that the gravity direction and an estimate of the camera height above the ground is known, [86] proposes a more complicated match consistency filter. For each 2D-3D correspondences, again obtained by matching image features against a SfM model, a set of camera pose hypotheses is generated. For each such pose, the 3D points in the model (including points that are non-matching) are projected into the query image. The projections are used to measure a semantic consistency score for the pose by counting the number of points projecting into a query image region with the same label as the point. The highest score from all poses of a match is then the semantic consistency score of that correspondence. The scores are normalized and used to bias RANSAC’s sampling strategy to prefer selecting more semantically consistent matches. While performing significantly better than SSMC [86], GSNC makes additional assumptions and is computationally less efficient.

**Particle Filter-based Semantic Localization (PFSL) [83].** In this approach, localization is approached as a filtering problem where we, in addition to a sequence of camera images, also have access to noisy odometry information. Both these sources are combined in a particle filter to sequentially estimate the pose of the camera by letting each particle describe a possible camera pose. In the update step of the particle filter, the new weight of each particle is set proportional to how well the projection of the 3D point cloud matches the segmentation of the current image. A 3D point p is assumed to match well if the pixel which p is projected to has the same label as p. Note that this approach does not depend on forming direct 2D-3D correspondences using, e.g., SIFT-descriptors, and is therefore more reliant on discriminative segmentation labels.

### 5. Experiments

The main focus of our experiments is evaluating the impact of using FGSNs for “semantic” visual localization. In addition, we investigate whether the clusters learned by the FGSNs carry semantic information.

**Network variations.** For training, we use two cross-season correspondence datasets from [42], namely the CMU Seasons Correspondence Dataset and the Oxford RobotCar Correspondence Dataset. The available samples are split into a training set (70% of the samples) and a validation set (30% of the samples). The corresponding images are geometrically separated from the query images in the Extended CMU Seasons and RobotCar Seasons benchmarks [74] used for evaluating the localization approaches.

In addition to comparing our results to several baselines, we investigate the impact of varying the number of output clusters as well as the impact of pretraining. For the latter, we evaluate a first variant that initializes the base of the network with weights from a network trained on ImageNet [24], while randomly initializing the rest of the network weights. A second variant uses a network pre-trained for semantic segmentation using the fine annotations of the Cityscapes dataset [22] and the training set of the Mapillary Vistas dataset [55]. To be able to combine these two datasets we mapped the Vistas semantic labels to the Cityscapes labels, hence 19 semantic classes were used during training.

Further, we train FGSNs with varying number of output clusters on Cityscapes and Vistas only. For these experiments $\mathcal{L}_{corr}$ was not used since there are no available correspondences for these datasets.

### 5.1. Semantic Information in Clusters

Our FGSNs are inspired by the task of semantic segmentation and designed with the goal of creating more fine-grained segmentations. Our training procedure does not enforce that our segmentations convey semantic information. Still, an interesting questions is whether our clusters can be related to standard semantic classes.

To investigate this, we calculate the normalized mutual information (NMI) to measure the shared information between cluster assignment and the semantic labels of the annotations in the Cityscapes [22] validation set. Denoting the cluster assignments as X and the semantic label assignments as Y, the normalized mutual information is given by

\[
NMI(X;Y) = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}},
\]

where I is the mutual information and H the entropy. If X and Y are independent, $NMI(X;Y) = 0$. If one of the assignments can be predicted from the other, then all information conveyed by X is shared with Y and $NMI(X;Y) = 1$. 

<table>
<thead>
<tr>
<th>Init Clusters</th>
<th>CMU</th>
<th>RobotCar</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CS</td>
<td>WD</td>
</tr>
<tr>
<td>Seg 20</td>
<td>40.1</td>
<td>33.7</td>
</tr>
<tr>
<td>Seg 100</td>
<td>47.0</td>
<td>36.6</td>
</tr>
<tr>
<td>Seg 200</td>
<td>47.0</td>
<td>36.6</td>
</tr>
<tr>
<td>Seg 1000</td>
<td>45.7</td>
<td>35.8</td>
</tr>
<tr>
<td>Class 200</td>
<td>28.8</td>
<td>26.7</td>
</tr>
<tr>
<td>Class 1000</td>
<td>18.1</td>
<td>22.2</td>
</tr>
</tbody>
</table>

Table 1. Measuring the semantic information contained in our clusters. Using models trained on the CMU or RobotCar Correspondence data, we measure the normalized mutual information (in %) between our clusters and the 19 Cityscapes classes on the Cityscapes (CS) and the WildDash (WD) validation sets. “Seg” networks are pre-trained on semantic segmentation and “Class” networks on classification.
Figure 3. Visualization of contingency tables between Cityscapes classes and cluster indices for different number of clusters. The clusters were trained on the CMU Correspondence Dataset using a model pre-trained on semantic segmentation. The colormap goes from dark blue (lowest value) to yellow (highest value). The data used is the 500 images from the Cityscapes validation set. Many of the classes common in the test images such as road, building and vegetation are split into several clusters.

In addition to the Cityscapes dataset, we also compare the cluster assignments with the same 19 classes on the WildDash dataset [96], which is designed to evaluate the robustness of segmentation methods under a wide range of conditions.

Tab. 1 shows the NMI for our networks. As expected, the networks pre-trained for semantic segmentation achieve a higher NMI compared to the networks pre-trained for classification. Intuitively, the clusters should thus contain semantic information that could be used for localization. However, a high NMI does not necessarily mean better localization performance. For example, a cluster containing pixels around the edges between house and sky would decrease the NMI between the cluster assignments and the semantic classes, but could be useful for localization.

Fig. 3 shows contingency tables between Cityscapes classes and our cluster indices for the networks trained on CMU with semantic segmentation initialization. Each contingency table displays the interrelation between two sets of assignments of the same data by forming a two-dimensional histogram, where each dimension corresponds to one of the assignments. In our case, the dimensions correspond to the semantic class labels and cluster indices, respectively. As can be seen, there are many cluster indices that are assigned to the same pixels as the semantic class vegetation. Since the CMU images contain a significant amount of vegetation, this is both expected and could lead to more information that can be used to localize the images. Looking at the contingency table for the network with 20 clusters, we can see that the cluster with index 19 overlaps with several of the semantic classes. This implies that many pixels are assigned to this cluster, indicating that semantic information is lost. This is also reflected in the NMI (c.f. Tab. 1), which is lower for 20-cluster networks compared to those trained with more clusters.

Fig. 3 also shows that many clusters do not directly correspond to semantic classes. This indicates that FGSNs deviate from the pre-trained networks used to initialize them.

5.2. Visual Localization

To verify that the learned clusters, even though they are not necessarily semantic in nature, contain useful information for visual localization, we perform experiments on two datasets for long-term visual localization: RobotCar Seasons [74] and the Extended CMU Seasons dataset [74].

Datasets. The RobotCar Seasons dataset consists of 32,792 images from the original RobotCar dataset [51]. Of these, 20,862 constitute a reference sequence with publicly known reference poses. A map triangulated from sparse features observed in these images is available as a reference 3D model as an aid for structure-based localization methods. The reference images are all captured under a single condition while the 11,934 test images are captured under a wide variety of different conditions, including seasonal, weather, and illumination changes. We use a slightly different version of the RobotCar Seasons dataset, also used in [42,74], which consists of a test and training set. The RobotCar Correspondences Dataset that we use to train our FGSNs overlaps with the training set, but not the test set of this version of the RobotCar Seasons dataset.

The Extended CMU Seasons dataset is a larger version of the CMU Seasons dataset from [74], based on the CMU Visual localization dataset [7]. Like the RobotCar Seasons dataset, the Extended CMU Seasons dataset consists of a reference sequence with publicly known camera poses, as well as a hidden test set whose camera poses are not publicly available. The reference sequence consists of 10,338 images captured during the same day in favourable conditions. The test set consists of 56,613 images captured during a wide variety of conditions (sunny, snowy, autumn, etc.). The dataset covers urban, suburban, and park-like areas dominated by vegetation on both sides of the road. The latter are the most challenging parts of this dataset [74].

Both datasets provide SIFT features for all test and training images. For SSMC and GSMC, we establish 2D-3D

3Available on visuallocalization.net.
Table 2. Localization performance for the SSMC method with different segmentation networks on the Extended CMU Seasons dataset and the RobotCar dataset. The first column marks entries from this paper, for the entry marked with * clustering was not repeated during training. Column two indicates the number of clusters (or classes) output by the network. Note that for entries marked with 19 and 66 use the semantic classes of Cityscapes and Vistas respectively and were trained with the method presented in [42]. Column three details what datasets were used during training: CS (Cityscapes), V (Vistas), E (Extra i.e. CMU for CMU results and RobotCar for RobotCar results), O (Other extra i.e. RobotCar for CMU results and RobotCar for RobotCar Results). The fourth column indicates, with ✓, if the correspondence loss was active during training while column five specifies the pretraining of the network (Seg for segmentation pretraining and Class for classification pretraining).

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Dataset</th>
<th>Urban / 250 / 0.5 / 5 [m]</th>
<th>Suburban / 250 / 0.5 / 5 [m]</th>
<th>Park / 250 / 0.5 / 5 [m]</th>
<th>Urban / 250 / 0.5 / 5 [deg]</th>
<th>Suburban / 250 / 0.5 / 5 [deg]</th>
<th>Park / 250 / 0.5 / 5 [deg]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS+V</td>
<td>✓</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS+V+E</td>
<td>✓</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS+V+O</td>
<td>✓</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS+V+O+E</td>
<td>✓</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 Cluster</td>
<td>Seg</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 Cluster</td>
<td>Seg</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 Cluster</td>
<td>Seg</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓ 100 Cluster</td>
<td>Seg</td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
<td>2.0 / 2 / 5 / 10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Comparison to state-of-the-art methods on the Extended CMU Seasons dataset. Best results for single-shot image localization and sequential localization are marked separately.

<table>
<thead>
<tr>
<th>Method / Setting</th>
<th>Urban 0.25 / 0.5 / 5 m</th>
<th>Urban 2 / 5 / 10</th>
<th>Suburban 0.25 / 0.5 / 5 m</th>
<th>Suburban 2 / 5 / 10</th>
<th>Park 0.25 / 0.5 / 5 m</th>
<th>Park 2 / 5 / 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSMC (FGSN, 100 clusters, trained on CMU)</td>
<td>85.3 / 91.0 / 94.6</td>
<td>69.5 / 76.4 / 82.7</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
</tr>
<tr>
<td>SSMC (FGSN, 200 clusters, trained on CMU)</td>
<td>85.3 / 91.0 / 94.6</td>
<td>69.5 / 76.4 / 82.7</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
<td>71.1 / 78.2 / 84.9</td>
</tr>
<tr>
<td>HF-Net [73]</td>
<td>89.5 / 94.2 / 97.9</td>
<td>87.6 / 87.2 / 91.0</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
</tr>
<tr>
<td>Asymmetric Hypercolumn Matching [29]</td>
<td>89.5 / 94.2 / 97.9</td>
<td>87.6 / 87.2 / 91.0</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
<td>84.3 / 89.4 / 93.2</td>
</tr>
<tr>
<td>GSCM [80]</td>
<td>90.7 / 95.3 / 98.7</td>
<td>89.6 / 91.4 / 94.5</td>
<td>85.5 / 89.7 / 94.9</td>
<td>85.5 / 89.7 / 94.9</td>
<td>85.5 / 89.7 / 94.9</td>
<td>85.5 / 89.7 / 94.9</td>
</tr>
</tbody>
</table>

Evaluation measures. We follow the evaluation protocol from [74] and report the percentage of query images localized within X meters and Y degrees of the ground-truth poses, using the same thresholds as in [74].

Impact of the number of clusters. In a first experiment, we evaluate the impact of the number of clusters learned by FGSNs on localization performance. For this experiment, we focus on the Simple Semantic Match Consistency (SSMC) and compare the performance of SSMC using FGSNs with varying numbers of clusters to the performance obtained with semantic segmentation algorithms. For the latter, we use networks jointly trained on Cityscapes and Vistas and on Cityscapes, Vistas, and the correspondence datasets [42], using the 19 Cityscapes classes and the 66 Vistas classes. Note that entries marked with ✓ also uses a correspondence loss similar to ours but for semantic classes.

Table 2 show the results of the experiments for the RobotCar and CMU datasets. As can be seen, using FGSNs trained with more than 20 clusters improves the localization performance. Especially under challenging conditions, i.e., night on RobotCar and Suburban and Park on CMU, the improvements obtained compared to semantic segmentations are substantial. Naturally, using too many clusters leads to an oversegmentation of the images and thus reduces the localization accuracy of SSMC. The experiments clearly show that SSMC benefits from using fine-grained segmentations, even though clusters might not necessarily match via descriptor matching [86]. Following [86], the Lowe ratio test with a threshold of 0.9 is used to filter out outliers. P3P RANSAC is then run for 10,000 iterations to estimate the camera pose.
correspond to standard semantic concepts.

The reason why SSMC benefits from a larger number of clusters is that the corresponding segmentations provide a more discriminative representation of the query images and the 3D point cloud. This allows SSMC to filter out more wrong matches by enforcing label consistency. This in turn increases the inlier ratio and thus the probability that RANSAC finds the correct pose. Plots detailing the impact of FGSNs with different numbers of clusters on the number of inliers and the inlier ratio are provided in the supplementary material.

According to Table 2, adding the Extra dataset decreases performance, this is most likely explained by the fact that the network had to be re-implemented to produce the results.

**Impact of pretraining FGSNs.** Entries marked with Class in column for of Table 2 show results obtained when pre-training the base networks of our FGSNs on a classification rather than a semantic segmentations task. As can be seen, FGSNs pre-trained on a classification task result in a significantly lower performance compared to networks trained for semantic segmentation. This shows the importance of using segmentations that retain some semantic information, which is more the case for FGSNs pre-trained on semantic segmentation than for FGSNs pre-trained on classification (c.f. Sec. 5.1).

**Impact of using 2D-2D point correspondences** Results for networks trained without the additional dataset from [42] or with the correspondence loss disabled (where the clustering still is done on feature from the CMU/RobotCar images), are shown in Table 2 (row 11-14).

As can be seen from the results, using fine-grained segmentation yields better results than using semantic classes on the Extended CMU Seasons dataset (c.f. entries CS+V (19 classes) and V (66 classes). These networks however, achieve lower results than their counterparts trained with the correspondence datasets. This indicates that the correspondence loss is important for localization performance.

**Generalization abilities.** Table 2 further show results obtained when training the FGSNs on a different dataset. We observe a substantial drop in performance compared to FGSNs trained on the same dataset. This behavior is not unexpected since the 2D-2D correspondences used to train our FGSNs encourage the network to learn dataset-specific clusters. While the performance of FGSNs trained on another dataset is comparable to using networks trained for semantic segmentation, our results indicate that there is still significant room for improving FGSNs.

**Repetition of clustering** Following the method developed by Caron et al. [15] the clustering is repeated after a set number of training iterations. Interestingly, we noticed that not resetting the network actually gives slightly better performance, see entry marked with * in Table 2. We attribute this to the network, pre-trained for semantic segmentation, retains semantic information more easily without resetting. Further investigation of this is left as future work.

**Comparison with state-of-the-art methods.** In a final experiment, we compare SSMC, GSMC and PFSL in combination with FGSNs to state-of-the-art on the Extended CMU Seasons dataset.

To this end, we compare against HF-Net [72], a CNN-based hierarchical localization approach, Asymmetric Hypercolumn Matching [29], an approach based on matching of hypercolumn features. DenseVLAD [87], a state-of-the-art image retrieval pipeline, and its trainable variant NetVLAD [3], City Scale Localization [84], a non-semantic approach based on 2D-3D matches, GSMC [86] using the semantic segmentation network from [86], and PFSL [83] using semantic segmentation network from [42].

As can be seen in Tab. 3, using segmentations with more labels, as afforded by our FGSNs, improves localization performance closing the performance gap to the current state-of-the-art. The results clearly validate the motivation behind FGSNs: using more segmentation labels to create more discriminative, yet still robust, representations for semantic visual localization.

## 6. Conclusion

In this paper, we have presented Fine-Grained Segmentation Networks (FGSN), a novel type of convolutional neural networks that output dense fine-grained segmentations. Using k-means clustering, we can train FGSNs in a self-supervised manner, using the cluster assignments of image features as labels. This enables us to use arbitrarily many output classes without having to create annotations manually. In addition, we have used a 2D-2D correspondence dataset [42] to ensure that the classes are stable under seasonal changes and viewpoint variations. Through extensive experiments, we have shown that using more fine-grained segmentations, as those of our FGSNs, is beneficial for the task of semantic visual localization.

Important future directions include further adapting visual localization methods to a larger number of clusters to ensure that the increased level of detail of the output segmentations is properly used. In addition, it would be interesting to further work on the generalization of FGSNs, e.g., in combination with domain adaptation methods.

**Acknowledgements** This work has been funded by the Swedish Research Council (grant no. 2016-04445), the Swedish Foundation for Strategic Research (Semantic Mapping and Visual Navigation for Smart Robots) and Vinnova / FFI (Perceptron, grant no. 2017-01942).
References


[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. PAMI, 2018. 2


[22] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016. 1, 2


[38] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Learned Contextual Feature Reweighting for Image Geo-Localization. In CVPR, 2017. 2
[67] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In CVPR, 2016. 3
[68] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. Target. 2