Learning Less is More – 6D Camera Localization via 3D Surface Regression

Eric Brachmann and Carsten Rother
Visual Learning Lab
Heidelberg University (HCI/IWR)
http://vislearn.de

Abstract

Popular research areas like autonomous driving and augmented reality have renewed the interest in image-based camera localization. In this work, we address the task of predicting the 6D camera pose from a single RGB image in a given 3D environment. With the advent of neural networks, previous works have either learned the entire camera localization process, or multiple components of a camera localization pipeline. Our key contribution is to demonstrate and explain that learning a single component of this pipeline is sufficient. This component is a fully convolutional neural network for densely regressing so-called scene coordinates, defining the correspondence between the input image and the 3D scene space. The neural network is prepended to a new end-to-end trainable pipeline. Our system is efficient, highly accurate, robust in training, and exhibits outstanding generalization capabilities. It exceeds state-of-the-art consistently on indoor and outdoor datasets. Interestingly, our approach surpasses existing techniques even without utilizing a 3D model of the scene during training, since the network is able to discover 3D scene geometry automatically, solely from single-view constraints.

1. Introduction

Precise localization is key in many applications of computer vision, such as inserting a virtual character into the smartphone view of your living room in a plausible way or guiding a self-driving car through New York. While different devices, like smartphones or self-driving cars, come with different sensors, an RGB camera is often present because of its low cost and rich output. In this work, we present a system which estimates the 6D camera pose, consisting of position and orientation, within a known 3D environment from a single RGB image.

In the last five years, we have seen an impressive leap forward of what computer vision can achieve due to advances in machine learning. Camera localization is a particularly difficult learning problem because training data is usually limited. Recording a dense or even regular sampling of 6D camera views for any given scene is impractical. Therefore, generalization to unseen views is a central capability of any camera localization system.
Because of scarce training data, learning the direct mapping of global image appearance to camera pose with a general purpose CNN (convolutional neural net) has seen only limited success. Approaches like PoseNet [11] and its derivatives [29, 10] exhibit low localization accuracy so far.

An alternative paradigm decomposes camera localization into a sequence of less complex tasks of which only some are learned. Recently, Brachmann et al. [2] presented the differentiable RANSAC (DSAC) pipeline for camera localization. It builds upon the scene coordinate regression framework originally proposed by Shotton et al. [23]. The main idea is to map image patches to corresponding points in 3D scene space, so called scene coordinates. This step can be learned even from limited data, since local patch appearance is relatively stable w.r.t. to view change. The camera pose, which aligns the image and predicted scene coordinates, can be estimated using RANSAC.

Specifically, in the case of DSAC [2], one CNN predicts scene coordinates, and then random subsets of scene coordinates are used to create a pool of camera pose hypotheses. Each hypothesis is scored by a second CNN (“scoring CNN”) according to its consensus with the global, i.e. image-wide, scene coordinate predictions. Based on these scores, one hypothesis is probabilistically chosen, refined and returned as the final camera pos estimate. The pipeline can be trained end-to-end by optimizing the expected loss of chosen hypotheses.

Brachmann et al. report state-of-the-art accuracy for indoor camera localization [2], but we see three main shortcomings of the DSAC pipeline. Firstly, the scoring CNN is prone to overfit because it can memorize global patterns of consensus to differentiate good from bad pose hypotheses. For example, the CNN might focus on where errors occur in the image rather than learning to assess the quality of errors. However, error location does not generalize well to unseen views. Secondly, initializing the pipeline for end-to-end training requires RGB-D training data or a 3D model of the scene to generate scene coordinate ground truth. Neither might be available in some application scenarios. Thirdly, end-to-end learning is unstable because in the DSAC pipeline [2] pose refinement is differentiated via finite differences which leads to high gradient variance.

In this work, we propose a new, fully differentiable camera localization pipeline which has only one learnable component, a fully convolutional neural net for scene coordinate regression. The output neurons of this network have a limited receptive field, preserving the patch-based nature of scene coordinate prediction. For hypothesis scoring, we utilize a soft inlier count instead of a learnable CNN. We show that this simple, differentiable scoring strategy is a reliable measure of pose quality, and yet impervious to overfitting. We present a new entropy control method to automatically adapt the magnitude of score values to ensure broad hypotheses distributions for stable end-to-end learning. We also deploy a new, analytical approximation of pose refinement gradients for additional training stability. Our pipeline is fast and substantially more accurate than state-of-the-art camera localization methods.

Additionally, and in contrast to previous works [23, 28, 9, 5, 3, 2], we explore learning scene coordinate regression without utilizing a 3D scene model or RGB-D training data. RGB-D data might not be available for outdoor scenes, and creating a scene reconstruction often requires tedious trial and error parameter search and manual corrections. Our system is able to discover an approximate scene geometry automatically due to a coarse initialization followed by optimization of scene coordinate reprojection errors. We can still utilize a 3D model if available but do not depend on it for accurate camera localization, see Fig. 1.

In the following, we summarize our main contributions.

- We present a new camera localization pipeline where a CNN regressing scene coordinates is the only learnable component. We implement hypothesis scoring with a new, entropy controlled soft inlier count without learnable parameters, which massively increases generalization capabilities.
- We show that pose refinement can be effectively differentiated using a local linearization which results in stable end-to-end learning.
- To the best of our knowledge, we are the first to show that scene coordinate regression can be learned using RGB images with associated ground truth poses, alone. Using a 3D model of the scene is optional since the system can discover scene geometry automatically.
- We improve accuracy of RGB-based 6D camera localization on three datasets, both indoor and outdoor, independent of training with or without a 3D model.

Related Work. Image-based localization has been addressed using image retrieval techniques, e.g. in [22] or more recently in [4, 1]. These methods match a query image to an image database annotated with pose information like geolocation. While these methods can scale to extremely large environments, they usually provide only a coarse estimate of the camera location.

Instead of matching to a database, Kendall et al. [11] proposed PoseNet, a CNN which learns to map an image directly to a 6D camera pose. The method has been improved in [10] by using a reprojection loss, and in [29] by using a more expressive architecture. Although accuracy increased somewhat in these recent works, they are still inferior to competing techniques discussed next.

Accurate 6D camera poses can be recovered using sparse feature-based pipelines. Matching local image descriptors to 3D points of a Structure-from-Motion scene reconstruction yields a set of 2D-3D correspondences, from which an accurate pose estimate can be recovered [13].
Research focused on making descriptor matching efficient [15], robust [25, 19] and scale to large outdoor environments [14, 18, 20]. However, local feature detectors rely on sufficiently textured scenes and good image quality [11]. Scene reconstruction can also be difficult for some environments [29] such as scenes with repeated texture elements. We surpass the accuracy of state-of-the-art feature-based methods for indoor and outdoor camera localization tasks, often even without using a 3D reconstruction.

The original scene coordinate regression pipeline for RGB-D camera localization of Shotton et al. [23] is related to sparse feature approaches by recovering camera pose by means of 2D-3D correspondences. But instead of matching a discrete set of points, Shotton et al. formulate correspondence prediction as a continuous regression problem. A random forest learns to map any image patch to a 3D scene point. The scene coordinate regression pipeline has been improved in several follow-up works, e.g. in terms of accuracy [28, 9] or learning camera localization on-the-fly [5]. However, these methods heavily depend on a depth channel which greatly simplifies the problem due to strong geometric constrains. An RGB version of the scene coordinate regression pipeline was proposed in [3] using an auto-context random forest. Similarly, Massiceti et al. [17] use a random forest mapped to a neural net within a similar pipeline. However, both systems can only be trained with scene coordinate ground truth, using either RGB-D data or a 3D scene model, and were not trained in an end-to-end fashion. Our approach is most closely related to the DSAC pipeline [2] which was introduced in detail above.

2. Method

Our pipeline follows the basic framework of differentiable RANSAC (DSAC) [2] which we describe in Sec. 2.1 for the task of camera pose estimation. In Sections 2.2 and 2.3, we explain the key architectural differences to the DSAC pipeline, namely using a fully convolutional network for scene coordinate regression, and scoring pose hypotheses using a soft inlier count without learnable parameters. See also Fig. 2 for an overview of our approach. We discuss the training procedure of our pipeline, with and without the use of a 3D scene model, in Sec. 2.4.

2.1. Background

Given an RGB image $I$, we aim at finding an estimate of the 6D camera pose $\hat{h}$ consisting of a 3D translation $\hat{t}$ and a 3D rotation $\hat{R}$. Our system has learnable parameters $w$ which control the search for pose estimate $h$. Differentiable RANSAC [2] estimates $\hat{h}$ in the following steps:

1. **Scene Coordinate Regression.** A CNN predicts for each pixel $i$ with position $p_i$ the corresponding 3D point $y_i(w)$ in the local coordinate frame of the scene. This scene coordinate $y_i(w)$ defines a 2D-3D correspondence between the image and the scene.

2. **Pose Hypothesis Sampling.** Four scene coordinates suffice to define a unique camera pose by solving the perspective-n-point problem [8]. Since predictions can be erroneous, a pool of $n$ pose hypotheses $h(w)$ is generated by selecting random 4-tupels of scene coordinate predictions. Each hypothesis $h(w)$ depends on parameters $w$ via the corresponding scene coordinates.

3. **Hypothesis Selection.** A function $s(h)$ scores the consensus of each hypothesis with all scene coordinate predictions. One hypothesis $h_j(w)$ with index $j$ is selected according to a probability distribution $P(j; w, \alpha)$ which is derived from the score values. Hypotheses with a high score are more likely to be selected. Hyper-parameter $\alpha$ controls the broadness of the distribution, and will be discussed together with the details of scoring in Sec. 2.3. Selecting hypothesis probabilistically facilitates end-to-end learning, as we will discuss shortly.
4. **Hypothesis Refinement.** Refinement $R$ is an iterative procedure which alternates between determining inlier pixels using the current pose estimate, and optimizing the estimate w.r.t. the inliers. We discuss the details of refinement in Sec. 2.4. The selected and refined pose hypothesis is the final estimate of the system, i.e. $h = R(h_2(w))$.

**Learning the Weights.** We assume a set of training images $\mathcal{D}$ with ground truth poses $h^*$. The probabilistic selection of a pose hypothesis in step 3 allows for optimizing learnable parameters $\mathbf{w}$ by minimizing the expected pose loss $\ell$ of the final estimate over the training set [2]:

$$\mathbf{w} = \arg\min_w \sum_{j} \mathbb{E}_{j \sim P(j; w, \alpha)} [\ell(R(h_j(w)), h^*)]. \quad (1)$$

Any differentiable loss $\ell(h, h^*)$ qualifies but we follow [2] by using $\ell(h, h^*) = \max(\angle(\theta, \theta^*), ||t - t^*||)$, i.e. the maximum of rotational and translational error. Partial derivatives of Eq. 1 w.r.t. parameters $\mathbf{w}$ are given by

$$\frac{\partial}{\partial \mathbf{w}} \mathbb{E}_j [\ell] = \mathbb{E}_j \left[ \frac{\partial}{\partial \mathbf{w}} \log P(j; \mathbf{w}, \alpha) + \frac{\partial}{\partial \mathbf{w}} \ell(\cdot) \right], \quad (2)$$

where we use $(\cdot)$ as a stand-in for corresponding arguments of Eq. 1. Eq. 2 allows us to learn our pipeline in an end-to-end fashion using standard back propagation.

2.2. Scene Coordinate Regression

The DSAC pipeline [2] uses a CNN for scene coordinate regression which takes an image patch of $42 \times 42$ px as input and produces one scene coordinate prediction for the center pixel. This design is inefficient because the CNN processes neighboring patches independently without reusing computations. They alleviate the problem by sampling $40 \times 40$ patches per image instead of making a dense prediction for all patches. In contrast, we use a fully convolutional network (FCN) [16], although without upsampling layers. Please see the supplementary materials for the exact architecture. Our FCN takes an RGB image of $640 \times 480$ px as input and produces $80 \times 60$ scene coordinate predictions, i.e. we regress more scene coordinates in less time. Similar to DSAC [2], we use a VGG-style [24] architecture with $\approx 30M$ parameters, and our output neurons have a receptive field of $41 \times 41$ px.

2.3. Hypothesis Scoring

Scoring determines which camera pose hypothesis is chosen and refined to yield the final estimate. DSAC [2] uses a separate CNN for this task. This scoring CNN takes a $40 \times 40$ image of reprojection errors, and regresses a score value $s(h)$ for each hypothesis. The reprojection error for pixel $i$ and hypothesis $h$ is defined as

$$r_i(h, \mathbf{w}) = ||Ch^{-1}y_i(\mathbf{w}) - p_i||, \quad (3)$$

where $C$ is the camera calibration matrix. We assume homogeneous coordinates, and application of perspective division before calculating the norm. The final hypothesis is chosen according to the softmax distribution $P(j; \mathbf{w}, \alpha)$:

$$j \sim P(j; \mathbf{w}, \alpha) = \frac{\exp(\alpha s(h_j(\mathbf{w})))}{\sum_k \exp(\alpha s(h_k(\mathbf{w})))}, \quad (4)$$

where hyper-parameter $\alpha$ is a fixed scaling factor that ensures a broad distribution. Parameter $\alpha$ controls the flow of gradients in end-to-end learning by limiting or enhancing the influence of hypotheses with smaller scores compared to hypotheses with larger scores.

We identify two problems when learning function $s(h)$. Firstly, the image of reprojection errors contains information about the global image structure, e.g. where errors occur. Scene coordinate regression generalizes well because only local, i.e. patch-based, predictions are being made. The scoring CNN of DSAC [2] however learns patterns in the global error image that do not generalize well to unseen views. Secondly, during end-to-end learning, the score CNN has an incentive to produce increasingly large scores which puts more weight on the best hypothesis over all other hypotheses in the pool. At some point, one hypothesis will have probability 1 resulting in the minimal expected task loss, see Eq. 1. Note that the scaling factor $\alpha$ is fixed in [2]. Hence, distribution $P(j; \mathbf{w}, \alpha)$ can collapse, leading to overfitting and training instability. Regularization might be able to alleviate both problems but we show that a simple and differentiable inlier counting schema is an effective and robust measure of camera pose quality, see Fig. 3.

**Soft Inlier Counting.** The original RANSAC-schema [6] measures hypothesis consensus by counting data points explained by the model, so called inliers. In our application, the inlier count is given by $\sum_i 1(\tau - r_i(h, \mathbf{w}))$, where $1$ denotes the Heaviside step function, $r_i$ denotes a reprojection error (see Eq. 3), and $\tau$ is the inlier threshold. Several earlier scene coordinate regression works relied on the inlier count to score hypotheses, e.g. [23, 3]. For our purpose, we construct a differentiable version by substituting the step function with a sigmoid function.

$$s(h) = \sum_i \sigma(\tau - \beta(r_i(h, \mathbf{w}))), \quad (5)$$

where hyper-parameter $\beta$ controls the softness of the sigmoid. This scoring function is similar to MSAC [26] but does not use a hard cut-off. We use Eq. 5 in distribution $P(j; \mathbf{w}, \alpha)$ to select a pose hypothesis.

**Controlling Entropy.** The magnitude of inlier scores can vary significantly depending on the difficulty of the scene, usually ranging from $10^2$ to $10^3$ for different environments. The magnitude can also change during end-to-end learning when scene coordinate regression improves and produces smaller reprojection errors. As mentioned earlier,
keeping scores within a reasonable range is important for having a broad distribution \( P(j; w, \alpha) \), and hence stabilizing end-to-end training. Setting \( \alpha \) manually per scene is a tedious task, hence we adapt \( \alpha \) automatically during end-to-end training. We measure distribution broadness via the Shannon entropy as a function of \( \alpha \):

\[
S(\alpha) = - \sum_j P(j; w, \alpha) \log P(j; w, \alpha). \tag{6}
\]

We optimize \( \alpha \) according to \( \arg\min_\alpha |S(\alpha) - S^*| \) with target entropy value \( S^* \) via gradient descent in parallel to end-to-end training of the pipeline. This schema establishes the target entropy within the first few iterations of end-to-end training and keeps it stable throughout.

### 2.4. Training Procedure

Our pipeline can be trained in an end-to-end fashion using pairs of RGB images and ground truth poses, but doing so from scratch will fail as the system quickly reaches a local minimum. The DSAC pipeline is initialized using scene coordinate ground truth extracted from RGB-D training data [2]. We propose a new 3-step training schema with different objective functions in each step. Depending on whether a 3D scene model is available or not, we use rendered or approximate scene coordinates to initialize the network in the first step. Training steps two and three improve the accuracy of the system which is crucial when no 3D model was provided for initialization.

**Scene Coordinate Initialization.** In the first training step, we initialize our pipeline similar to DSAC [2] by optimizing

\[
\hat{w} = \arg\min_w \sum_i ||y_i(w) - y_i^*||. \tag{7}
\]

We render scene coordinates \( y^* \) using ground truth poses \( h^* \), and a 3D scene model, if available. Without a 3D model, we approximate scene coordinate \( y^*_i \) by \( h^*(\frac{dx_i}{f}, \frac{dy_i}{f}, d, 1)^T \), where \( x_i \) and \( y_i \) are the 2D coordinates of pixel \( i \), \( f \) denotes the focal length, and \( d \) represents a constant depth prior. This heuristic assumes that all scene points have a constant distance from the camera plane, see Fig. 4 for a visualization. The heuristic ignores scene geometry completely. However, it effectively disentangles camera views by coarsely assigning the correct range of scene coordinate values to different spatial parts of the 3D environment. The heuristic itself will yield poor localization accuracy, but serves as basis for the next training step.

**Optimization of Reprojection Error.** In a second training step, we optimize the reprojection error, calculated using the ground truth pose. It encodes single view constraints that are effective for recovering the correct depth of scene points in case we used the heuristic in the first training step. Thus, we optimize

\[
\hat{w} = \arg\min_w \sum_i r_i(h^*, w), \tag{8}
\]

which lets our system learn about the scene geometry without using a 3D scene model. However, we found that optimizing the reprojection error can improve accuracy, even if a 3D model was available for the initialization.
End-to-End Optimization. In a third step, we train our pipeline in an end-to-end fashion according to Eq. 1. This requires all components to be differentiable, including pose refinement. In DSAC [2], refinement gradients are calculated via finite differences. Note that refinement depends on thousands of inputs, namely all scene coordinate predictions, which makes calculation of finite differences slow, and results in high gradient variance due to numerical instabilities. In the following, we discuss pose refinement in detail, and explain an efficient, analytical approximation for refinement gradients, resulting in stable end-to-end training.

As mentioned earlier, pose refinement alternates between, firstly, determining a set of inlier pixels w.r.t. to the current pose estimate, and, secondly, optimizing the pose w.r.t. to reprojection errors over the inlier set. We define the inlier set \( \mathcal{I} \) to contain all pixels with a reprojection error \( r_i \) below a threshold \( \tau \), i.e. \( \mathcal{I} = \{ i | r_i < \tau \} \). For pose optimization, we combine the reprojection errors of all inliers within one residual vector:

\[
(r'(h, w))_i = \begin{cases} 
  r_i(h, w) & \text{if } i \in \mathcal{I}, \\
  0 & \text{otherwise}
\end{cases} 
\]  
(9)

We optimize the pose according to:

\[
R(h) = \text{argmin}_{h'} \| r(h', w) \|^2 . 
\]  
(10)

We use the iterative Gauss-Newton algorithm, which gives the following update rule:

\[
R^{t+1} = R^t - (J_r^T J_r)^{-1} J_r^T r(R^t, w),
\]  
(11)

where \( t \) denotes the iteration number, and we abbreviate \( R^t(h) \) by \( R^t \). The Jacobian matrix \( J_r \) contains partial derivatives \((J_r)_{ij} = \frac{\partial r_i(R^t, w)}{\partial w_j})\). We optimize until convergence, re-calculate the inlier set, and repeat until the inlier set converges, too. Note that for DSAC [2] the number of refinement iterations and the number of inliers considered are limited to reduce the computational demand of finite differences. We do not have to make similar concessions, here.

The model linearization of Gauss-Newton allows us to approximate refinement gradients \( \frac{\partial}{\partial w} R(h) \) around the optimum found in the last optimization iteration [7]. We fix the optimum as \( h_0 = R^{t=\infty}(h) \) which makes the last update step \( R(h) = h_0 - (J_r^T J_r)^{-1} J_r^T r(h_0, w) \), and the corresponding derivatives

\[
\frac{\partial}{\partial w} R(h) \approx -(J_r^T J_r)^{-1} J_r^T \frac{\partial}{\partial w} r(h_0, w),
\]  
(12)

which allow for stable end-to-end training according to Eq. 1.

3. Experiments

We evaluate our approach on three publicly available camera localization datasets, both indoor and outdoor. 

7Scenes [23]. This RGB-D dataset comprises of 7 difficult indoor scenes with motion blur, repeating structures and texture-less surfaces. Several thousand frames are given as training and test sequences for each scene. The dataset includes ground truth poses and accurate 3D scene models. We ignore depth channels and utilize only RGB images.

12Scenes [27]. This dataset is very similar to 7Scenes but has larger indoor environments, and contains smaller training sets of several hundred frames per scene.

Cambridge Landmarks [11]. The dataset contains RGB images of six large outdoor environments, divided in training and test sequences of several hundred frames. Coarse SfM reconstructions are also provided.

Parameter Settings. We train our pipeline for a fixed number of iterations using ADAM [12] on full training sets, and select hyper-parameters that achieve the lowest training loss. During test time we always choose the hypothesis with maximum score. See the supplementary materials for a full parameter listing.

Competitors. We compare to the latest incarnation of PoseNet [10] which can be trained using a standard pose loss or, utilizing a 3D model, a geometric loss for improved accuracy. We also compare to the Spatial LSTM of [29]. We include results of several sparse feature baselines, most notably Active Search [20]. We compare to DSAC [2] trained using RGB-D training data as in [2], and using rendered scene coordinates (denoted “RGB training”).

3.1. Camera Localization Accuracy

We list our main experimental results in Table 1 for the 7Scenes and Cambridge datasets. Compared to the PoseNet variants [10, 29] we improve accuracy by a factor of 10 for many scenes, and compared to the sparse feature-based Active Search [20] by a factor of 2. Compared to DSAC [2], which is the strongest competitor, we massively improve accuracy for the Cambridge dataset. We observe only a small to moderate loss in accuracy when our method is trained without a 3D scene model. Note that the only two competitors that do not depend on a 3D model, namely PoseNet [10] and the spatial LSTM [29], achieve a much lower accuracy. In fact, for most scenes, our method trained without a 3D model surpasses the accuracy of competitors utilizing a 3D model. Similar to DSAC [2] and the Spatial LSTM [29], we were not able to estimate reasonable poses for the Cambridge Street scene which is one order of magnitude larger than the other outdoor scenes. The capacity of our neural network might be insufficient for this scene scale but we did not explore this possibility.

The median pose accuracy used in Table 1 does not reflect the probability of wrong pose estimates very well.
**Table 1. Median 6D Localization Errors.** We report results for the 7Scenes dataset [23] and the Cambridge Landmarks dataset [11]. We mark best results **bold** (if both, translational and rotational error, are lowest). Results of DSAC marked with an asterisk (*) are before end-to-end optimization which did not converge. A dash (-) indicates that a method failed completely.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training w/ 3D Model</th>
<th>w/o 3D Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>7Scenes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chess</td>
<td>0.13m, 4.5°</td>
<td>0.14m, 4.5°</td>
</tr>
<tr>
<td>Fire</td>
<td>0.27m, 11.3°</td>
<td>0.27m, 11.8°</td>
</tr>
<tr>
<td>Heads</td>
<td>0.17m, 13.0°</td>
<td>0.18m, 12.1°</td>
</tr>
<tr>
<td>Office</td>
<td>0.19m, 5.6°</td>
<td>0.20m, 5.8°</td>
</tr>
<tr>
<td>Pumpkin</td>
<td>0.26m, 4.8°</td>
<td>0.25m, 4.8°</td>
</tr>
<tr>
<td>Kitchen</td>
<td>0.23m, 5.4°</td>
<td>0.24m, 5.5°</td>
</tr>
<tr>
<td>Stairs</td>
<td>0.35m, 12.4°</td>
<td>0.37m, 10.6°</td>
</tr>
<tr>
<td>Great Court</td>
<td>7.00m, 3.7°</td>
<td>6.83m, 3.5°</td>
</tr>
<tr>
<td>K. College</td>
<td>0.99m, 1.1°</td>
<td>0.88m, 1.0°</td>
</tr>
<tr>
<td>Old Hospital</td>
<td>2.17m, 2.9°</td>
<td>3.20m, 3.3°</td>
</tr>
<tr>
<td>Shop Facade</td>
<td>1.05m, 4.0°</td>
<td>0.88m, 3.8°</td>
</tr>
<tr>
<td>St. M. Church</td>
<td>1.49m, 3.4°</td>
<td>1.57m, 3.2°</td>
</tr>
<tr>
<td>Street</td>
<td>20.7m, 25.7°</td>
<td><strong>20.3m, 25.5°</strong></td>
</tr>
</tbody>
</table>

Figure 5. **Indoor Localization Accuracy.** We show the percentage of test frames of the 7Scenes dataset with a pose error below 5cm and 5°. We mark our method blue. Note that all competitors depend on a 3D model. For our method, we show results after each of our 3 training steps on the 7Scenes dataset.

We retrain DSAC, substituting the scoring CNN for our soft inlier count. We measure the percentage of test frames with an error below 5cm and 5°. Results improve from 55.9% to 58.9% for 7Scenes. The effect is strongest for the **Heads** and **Stairs** scenes (+19% resp. +8%) which have the smallest training sets. Accuracy for 12Scenes, where all training sets are small, also increases from 79.7% to 89.6%. We conclude that the soft inlier count helps generalization, considerably.

---

**3.2. Detailed Studies**

**Inlier Count vs. Scoring CNN.** We retrain DSAC, substituting the scoring CNN for our soft inlier count. We measure the percentage of test frames with an error below 5cm and 5°. Results improve from 55.9% to 58.9% for 7Scenes. The effect is strongest for the **Heads** and **Stairs** scenes (+19% resp. +8%) which have the smallest training sets. Accuracy for 12Scenes, where all training sets are small, also increases from 79.7% to 89.6%. We conclude that the soft inlier count helps generalization, considerably.

---

**Figure 5.** Indoor Localization Accuracy. We show the percentage of test frames of the 7Scenes (a) resp. the 12Scenes (b) dataset with a pose error below 5cm and 5°. We mark our method blue. Note that all competitors depend on a 3D model. For our method, we show results after each of our 3 training steps on the 7Scenes dataset (c).

Therefore, we show the percentage of test images with a pose error below 5cm and 5° for the 7Scenes dataset in Fig. 5 a). Note that all competitors listed require a 3D model of the scene. PoseNet [10] and the Spatial LSTM [29] do not report results using this measure, but based on their median accuracy they achieve less than 50% on 7Scenes. We outperform all competitors, most notably the RGB-D [2] trained with RGB-D data (+13.6%). When training DSAC using a 3D model (“RGB Training”), its performance drops by 6.6% due to inaccuracies in the 3D model. Our method trained without a 3D model exceeds the accuracy of DSAC trained with a 3D model by 4.5%.

We show results for the 12Scenes dataset in Fig. 5 b). We achieve an accuracy of 96.4% with a 16.7% margin to DSAC. Training without a 3D model still achieves a good accuracy comparable to a sparse feature baseline (SIFT+PNP [27]). See Fig. 6 for a qualitative comparison of DSAC and our method trained with and without a 3D model, respectively. We include additional qualitative results in the supplementary video.1

---

1https://youtu.be/DjJFRTFEuQo
Figure 6. **Estimated Camera Trajectories.** We plot estimated camera locations as trajectories within the respective 3D scene model (untextured for 7Scenes). We show ground truth in green, DSAC [2] estimates in orange, and our results in blue and cyan when trained with and without a 3D model, respectively. Note that DSAC produces many wrong estimates despite being trained with a 3D model.

The scoring CNN easily overfits to the spatial constellation of reprojection errors. For large training sets (e.g. 7Scenes Kitchen) accuracy differs by less than 1% between the two scoring methods, but inlier counting is always superior. We expect the accuracy gap to vanish completely given enough training data, but at the moment we see no evidence that the scoring CNN could learn to make a more intelligent decision than the inlier counting schema.

**Impact of Training Steps.** See Fig. 5 c) for a breakdown of our 3-step training procedure. When a 3D scene model is available for training, the first training step already achieves high accuracy. Optimizing the reprojection error and end-to-end training improve accuracy further. When no 3D model is available, optimizing the reprojection error is imperative for good results, since discovering scene geometry is necessary to generalize to unseen views. End-to-end training can additionally improve accuracy but effects are small. We observed that end-to-end training alone is insufficient to recover from the heuristic initialization.

**Stability of End-to-End Training.** For three out of six Cambridge scenes, DSAC’s end-to-end training did not converge despite manually tuning training parameters. In contrast, end-to-end training of our method converges for all scenes due to learning with broad hypothesis distributions, and our improved approximation of refinement gradients.

**Additional Analysis.** Please see the supplementary materials for additional experimental analysis. This includes detailed quantitative results for 7Scenes and 12Scenes, the impact of the FCN output resolution on accuracy, as well as the impact of our heuristic scene coordinate initialization. Furthermore, we visualize scene geometry discovered automatically by the FCN, and comment on running times of our pipeline.

### 4. Conclusion

We have presented a method for 6D camera pose estimation from RGB images which surpasses accuracy of sparse feature-based and learning-based competitors. The system can learn from small training sets, and still generalize well to unseen views. Training can utilize a 3D scene model, or discover scene geometry automatically. We make the source code and trained models publicly available\(^2\).

Our method scales to large outdoor scenes but fails on city-scale scenes like the challenging Cambridge Street scene [11]. In the future, we could pair our system with an image retrieval approach as proposed in [21].

**Acknowledgements:** This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 647769). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden. We thank Tobias Witt for his help visualizing our results, and Lynton Ardizzone for inspiring discussions on the differentiability of iterative optimization.

\(^2\)[https://hci.iwr.uni-heidelberg.de/vislearn/research/scene-understanding/pose-estimation/#CVPR18]
References


