This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

Visual Camera Re-Localization from RGB and RGB-D Images Using DSAC

Eric Brachmann and Carsten Rother

Abstract—We describe a learning-based system that estimates the camera position and orientation from a single input image relative to a known environment. The system is flexible w.r.t. the amount of information available at test and at training time, catering to different applications. Input images can be RGB-D or RGB, and a 3D model of the environment can be utilized for training but is not necessary. In the minimal case, our system requires only RGB images and ground truth poses at training time, and it requires only a single RGB image at test time. The framework consists of a deep neural network and fully differentiable pose optimization. The neural network predicts so called scene coordinates, i.e. dense correspondences between the input image and 3D scene space of the environment. The pose optimization implements robust fitting of pose parameters using differentiable RANSAC (*DSAC*) to facilitate end-to-end training. The system, an extension of DSAC++ and referred to as DSAC*, achieves state-of-the-art accuracy on various public datasets for RGB-based re-localization, and competitive accuracy for RGB-D based re-localization.

Index Terms—Camera Re-Localization, Pose Estimation, Differentiable RANSAC, DSAC, Differentiable Argmax, Differentiable PnP

1 Introduction

The ability to re-localize ourselves has revolutionized our daily lives. GPS-enabled smart phones already facilitate car navigation without a co-driver sweating over giant fold-out maps, or they enable the search for a rare vegetarian restaurant in the urban jungle of Seoul. On the other hand, the limits of GPS-based re-localization are clear to anyone getting lost in vast indoor spaces or in between sky scrapers. When the satellite signals are blocked or delayed, GPS does not work or becomes inaccurate. At the same time, upcoming technical marvels, like autonomous driving [1] or impending updates of reality itself (i.e. augmented/extended/virtual reality [2]), call for reliable, high precision estimates of camera position and orientation.

Visual camera re-localization systems offer a viable alternative to GPS by matching an image of the current environment, e.g. taken by a handheld device, with a database representation of said environment. From a single image, state-of-the-art visual re-localization methods estimate the camera position to the centimeter, and the camera orientation up to a fraction of a degree, both indoors and outdoors.

Existing re-localization approaches rely on varying types of information to solve the task, effectively catering to different application scenarios. Some use RGB-D images as input which facilitates highest precision suitable for augmented reality [3], [4], [5], [6]. However, they require capturing devices with active or passive stereo capabilities, where the former only works indoors, and the latter requires a large stereo baseline for reliable depth estimates outdoors. Approaches based on feature-matching use an RGB image as input and also offer high precision [7]. But they require a structure-from-motion (SfM) reconstruction [8], [9], [10] of the environment for re-localization. Such reconstructions might be cumbersome to obtain indoors due to textureless surfaces and repeating structures obstructing reliable feature matching [11]. Finally, approaches based on image retrieval or pose regression require only a database of RGB images and ground truth poses for re-localization, but suffer from low precision comparable to GPS [12].

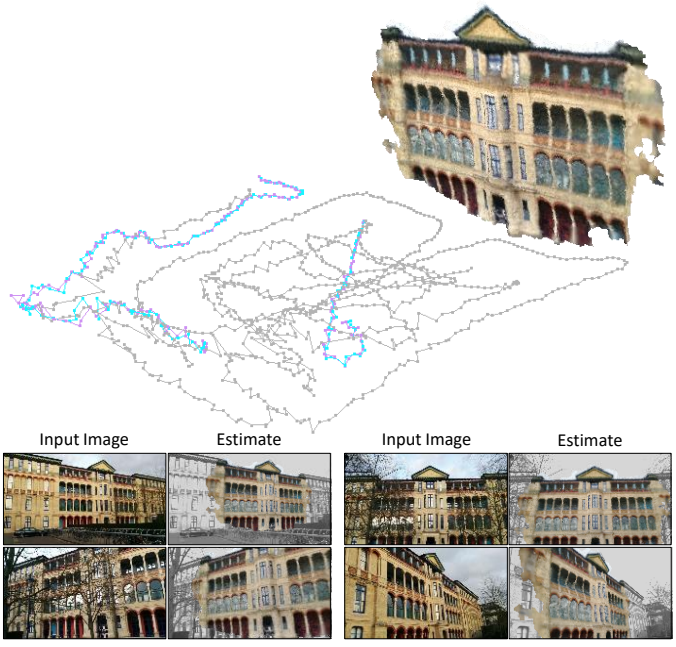


Fig. 1. **Top.** Our system accurately re-localizes within a known environment given a single image. We show estimated camera positions in purple and ground truth in cyan. In this instance, the system was trained using RGB images and associated ground truth poses, only (gray trajectory), In particular, the scene geometry, displayed as a 3D model, was discovered by the system, automatically. **Bottom.** To visualize the re-localization quality, we render the learned 3D geometry using estimated poses over gray-scale input images.

In this work, we describe a versatile, learning-based framework for visual camera re-localization that covers all aforementioned scenarios. In the minimal case, it requires only a database of RGB images and ground truth poses of an environment for training, and re-localizes based on a single RGB image at test time with high precision. In such a scenario the system automatically discovers the 3D geometry of the environment during training, see Fig. 1 for

E. Brachmann is with Niantic, Inc. Work done during his time at the Visual Learning Lab, Heidelberg University.
C. Rother is with the Visual Learning Lab, Heidelberg University.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

an example. If a 3D model of the scene exists, either as a SfM reconstruction or a 3D scan, we can utilize it to help the training process. The framework exploits depth information at training or test time if an RGB-D sensor is available.

We base our approach on scene coordinate regression initially proposed by Shotton et al. [3] for RGB-D-based camera re-localization. A learnable function, a random forest in [3], regresses for each pixel of an input image the corresponding 3D coordinate in the environment's reference frame. This induces a dense correspondence field between the image and the 3D scene that serves as basis for RANSAC-based pose optimization. In our work, we replace the random forest of [3] with a fully convolutional neural network [13], and derive differentiable approximations to all steps of pose optimization. Most prominently, we derive a differentiable approximation of the RANSAC robust estimator, called *differentiable sample consensus* (DSAC) [14]. Additionally, we describe an efficient differentiable approximation for calculating gradients of the perspective-n-point problem [15]. These ingredients make our framework end-to-end trainable, ensuring that the neural network predicts scene coordinates that result in high precision camera poses.

This article is a summary and extension of our previous work on camera re-localization published in [14] as DSAC, and its follow-up DSAC++ [15]. In particular, we describe an improved version under the name *DSAC** with the following properties.

- We extend DSAC++ to optionally utilize RGB-D inputs. The corresponding pose solver is naturally differentiable, and other components require only minor adjustments. When using RGB-D, DSAC* achieves accuracy comparable to the state-of-the-art on standard indoor re-localization datasets.
- We propose a simplified training procedure which unifies the two separate initialization steps used in DSAC++. As a result, DSAC* needs less than half the training time of DSAC++ on identical hardware.
- The improved initialization also leads to better accuracy. Particularly, when training without a 3D model, results improve significantly from 53.1% (DSAC++) to 80.7% (DSAC*) for indoor re-localization.
- We utilize an improved network architecture for scene coordinate regression which we introduced in [16], [17]. The architecture, based on ResNet [18], reduces the memory footprint by 75% compared to the network of DSAC++. The new architecture is also considerably faster, and together with better pose optimization parameters we reduce inference time of DSAC* to less than 40% compared to DSAC++ on identical hardware.
- In new ablation studies, we investigate the impact of training data augmentation, the impact of the network's receptive field, as well as the impact of end-to-end training. We also analyze the scene compression properties of DSAC*. Furthermore, we provide extensive visualizations of our pose estimates, and of the 3D geometry that the network encodes.
- We migrate our implementation of DSAC++ from LUA/Torch to PyTorch [19] and make it publicly available: https://github.com/vislearn/dsacstar

This article is organized as follows: We give an overview of related work in Sec. 2. In Sec. 3, we formally introduce the task of camera re-localization and how we solve it via scene coordinate regression. In Sec. 4, we discuss how to train the scene coordinate network using auxiliary losses defined on the scene coordinate output. In Sec. 5, we discuss how to train the whole system end-to-end, optimizing a loss on the estimated camera pose. We present experiments for indoor and outdoor camera re-localization, including ablation studies in Sec. 6. We conclude this article in Sec. 7.

2 Related Work

In the following, we discuss the main strains of research for solving visual camera re-localization. We also discuss related work on differentiable robust estimators other than DSAC.

2.1 Image Retrieval and Pose Regression

Early examples of visual re-localization rely on efficient image retrieval [20]. The environment is represented as a collection of database images with known camera poses. Given a query image, we search for the most similar database image by matching global image descriptors, such as DenseVLAD [21], or its learned successor NetVLAD [22]. The metric to compare global descriptors can be learned as well [23]. The sampling density of database images inherently limits the accuracy of retrieval-based system. However, they scale to very large environments, and can serve as an efficient initialization for local pose refinement [24], [25].

Absolute pose regression methods [11], [26], [27], [28], [29] aim at overcoming the precision limitation of image retrieval while preserving efficiency and scalability. Interpreting the database images as a training set, a neural network learns the relationship between image content and camera pose. In theory, the network could learn to interpolate poses of training images, or even generalize to novel view points. In practise, however, absolute pose regression fails to consistently outperform the accuracy of image retrieval methods [12].

Relative pose regression methods [30], [31] train a neural network to predict the relative transformation between the query image, and the most similar database image found by image retrieval. Initial relative pose regression methods suffered from similarly low accuracy as absolute pose regression [12]. However, recent work [32] suggests that relative pose regression can achieve accuracy comparable to structure-based methods which we discuss next.

2.2 Sparse Feature Matching

The camera pose can be recovered by matching sparse, local features like SIFT [33] between the query image and database images [34]. For an efficient database representation, SfM tools [9], [10] create a sparse 3D point cloud of an environment, where each 3D point has one or several feature descriptors attached to it. Given a query image, feature matching established 2D-3D correspondences which can be utilized in RANSAC-based pose optimization to yield a very

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

precise camera pose estimate [7]. Work on feature-based re-localization has primarily focused on scaling to very large environments [24], [35], [36], [37], [38], [39] enabling city or even world scale re-localization. Other authors worked on efficiency to run feature-based re-localization on mobile devices with low computational budget [40].

While sparse feature matching can achieve high re-localization accuracy, hand-crafted features fail in certain scenarios. Feature detectors have difficulty finding stable points under motion blur [26] and for texture-less areas [3]. Also, SfM reconstructions tend to fail in indoor environments dominated by ambiguous, repeating structures [11]. Learning-based sparse feature pipelines [41], [42], [43], [44] might ultimately be able to overcome these issues, but currently it is an open research question whether learned sparse features consistently exceed the capabilities of their hand-crafted predecessors [45], [46].

State-of-the-art feature-based re-localization methods such as ActiveSearch [7] offer no direct possibility to incorporate depth sensors when available at test time, neither do current state-of-the-art SfM tools like COLMAP [10] support depth sensors when creating the scene reconstruction.

2.3 Scene Coordinate Regression

Instead of relying on a feature detector to identify salient image structures suitable for discrete matching, scene coordinate regression [3] predicts the corresponding 3D scene point for a given 2D pixel location, directly. In these works, the environment is implicitly represented by a learnable function that can be evaluated for any image pixel to predict a dense correspondence field between image and scene. The correspondences serve as input for RANSAC-based pose optimization, similar to sparse feature techniques.

Originally, scene coordinate regression was proposed for RGB-D-based re-localization in indoor environments [3], [4], [47], [48]. The depth channel would serve as additional input to a scene coordinate regression forest, and be used in pose optimization by allowing to establish and resolve 3D-3D correspondences [49]. Scene coordinate regression forests were later shown to also work well for RGB-based re-localization [5], [50].

Recent works on scene coordinate regression often replace the random forest regressor by a neural network while continuing to focus on RGB inputs [14], [15], [17], [51], [52]. In previous work, we have shown that the RANSAC-based pose optimization can be made differentiable to allow for end-to-end training of a scene coordinate regression pipeline [14], [15]. In particular, [14] introduced a differentiable approximation of RANSAC [53], and [15] described an efficient analytical approximation of calculating gradients for perspective-n-point solvers. Furthermore, the predecessor of the current work, DSAC++ [15], introduced the possibility to train scene coordinate regression solely from RGB images and ground truth poses, without the need for image depth or a 3D model of the scene. Li et al. [52] improved on this initial effort by enforcing multi-view and photometric consistency throughout training. In a follow-up work, Li et al. [54] introduce a joint classification-regression network architecture for predicting scene coordinate, and demonstrate the effectiveness of training data augmentation for large improvements on standard benchmarks.

In this work, we describe several improvements to DSAC++ that increase accuracy while reducing training and test time. We demonstrate that the DSAC framework naturally exploits image depth if available, in an attempt to unify previously distinct strains of RGB- and RGB-D-based re-localization research. In summary, our method is more precise and more flexible than previous scene coordinate regression- and sparse feature-based re-localization systems. At the same time, it is as simple to deploy as absolute pose regression systems due to requiring only a set of RGB images with ground truth poses for training in the minimal setting.

Orthogonal to this work, we describe a scalable variant of DSAC-based re-localization in [17]. Yang et al. explore the possibility of allowing for scene-independent coordinate regression [55], and Cavallari et al. adapt scene coordinate regression forests and networks on-the-fly for deployment as a re-localizer in simultaneous localization and mapping (SLAM) [56], [57], [58].

2.4 Differentiable Robust Estimators

To allow for end-to-end training of our re-localization pipeline, we have introduced a differentiable approximation to the RANSAC [53] algorithm, called *differentiable sample consensus* (DSAC). DSAC relies on a formulation of RANSAC that reduces to an argmax operation over model parameters. Instead of choosing model parameters with maximum consensus, we choose model parameters randomly with a probability *proportional to consensus*. This allows us to optimize the expected task loss for end-to-end training. A DSAC variant using a soft argmax [59] does not work as well since it ignores potential multi-modality in the distribution of model parameters. Recently, Lee et al. proposed a kernel soft argmax as an alternative that is robust to multiple modes in the arguments [60]. However, their approximation effectively suppresses gradients of all but the main mode, while the DSAC estimator utilizes gradients of all modes.

Alternatively to making RANSAC differentiable, some authors propose to replace RANSAC by a neural network [61], [62], [63], [64], [65]. In these works, the neural network acts as a classifier for model inliers, effectively acting as a robust estimator for model parameters. However, *NG-RANSAC* [16] demonstrates that the *combination* of an inlier-scoring network and RANSAC achieves even higher accuracy. In [16], we also discuss a combination of NG-RANSAC and DSAC for camera re-localization which leads to higher accuracy in outdoor re-localization by learning to focus on informative image areas.

3 Framework

In this section, we introduce the task of camera re-localization, and the principle of scene coordinate regression [3]. We explain how to estimate the camera pose from scene coordinates using RANSAC [53] when the input is a single RGB or RGB-D image, respectively.

Given an image I, which can be either RGB or RGB-D, we aim at estimating camera pose parameters $\mathbf{h}$ w.r.t. the reference coordinate frame of a known scene, a task called

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

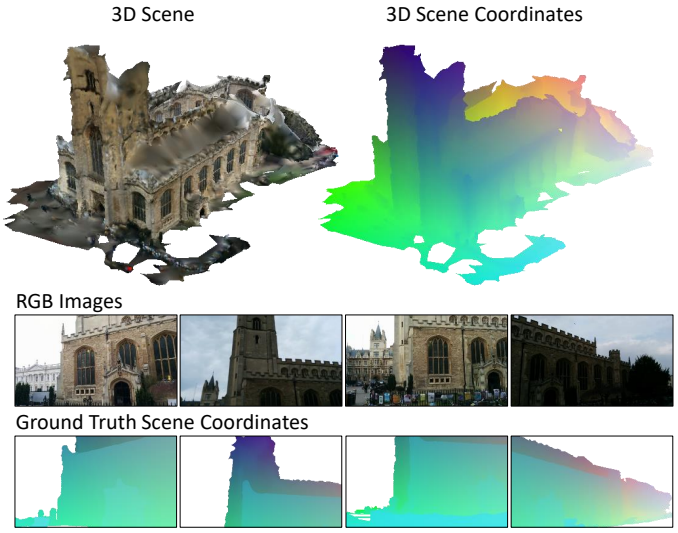


Fig. 2. **Scene Coordinates [3]. Top.** Every surface point in a 3D environment has a unique 3D coordinate in the local coordinate frame. We visualize 3D scene coordinates by mapping XYZ to the RGB cube. **Bottom.** A 3D scene model together with ground truth camera poses allows us to render ground truth scene coordinates for images, e.g. to serve as training targets. We can also create training targets from depth maps instead of a 3D model, or from ground truth poses alone by optimizing the re-projection error over multiple frames.

re-localization. We propose a learnable system to solve the task, which is trained for a specific scene to re-localize within that scene. The camera pose has 6 degrees of freedom (DoF) corresponding to the 3D camera position $\mathbf{t}$ and its 3D orientation $\boldsymbol{\theta}$. In particular, we define the camera pose as the transformation that maps 3D points in the camera coordinate space, denoted as $\mathbf{e}$ to 3D points in scene coordinate space, denoted as $\mathbf{y}$, i.e.

$$\mathbf{y}_i = \mathbf{h}\mathbf{e}_i, \tag{1}$$

where i denotes the pixel index in image I. For notational simplicity, we assume a 4x4 matrix representation of the camera pose $\mathbf{h}$ and homogeneous coordinates for all points where convenient.

We denote the complete set of scene coordinates for a given image as $\mathcal{Y}$, i.e. $\mathbf{y}_i \in \mathcal{Y}$. See Fig. 2 for an explanation and visualization of scene coordinates. Originally proposed by Shotton et al. [3], scene coordinates $\mathcal{Y}$ induce a dense correspondence field between camera coordinate space and scene coordinate space which we can use to solve for the camera pose. To estimate $\mathcal{Y}$ for a given image, we utilize a neural network f with learnable parameters $\mathbf{w}$:

$$\mathcal{Y} = f(I; \mathbf{w}). \tag{2}$$

Due to potential errors in the neural network prediction, we utilize a robust estimator, namely RANSAC [53], to recover $\mathbf{h}$ from $\mathcal{Y}$. Our RANSAC-based pose optimization consists of the following steps:

1) Sample a set of camera pose hypotheses.
2) Score each hypothesis and choose the best one.
3) Refine the winning hypothesis.

We show an overview of our system in Fig. 3. In the following, we describe the three aforementioned steps for the general case, while we elaborate on concrete manifestations for RGB and RGB-D input images in Sec. 3.1 and Sec. 3.2, respectively.

1) Sample Hypotheses. Image I and scene coordinate prediction $\mathcal{Y}$ define a dense correspondence field $\mathcal{C}$ over all image pixels i. We will specify the concrete nature of correspondences in sub-sections below because it differs for RGB and RGB-D inputs. As the first step of robust pose optimization we randomly choose M minimal subsets of correspondences, $\mathcal{C}_j \subseteq \mathcal{C}$, with $0 \leq j < M$. We assume minimal subsets to contain the smallest number of correspondences necessary to fit model parameters without ambiguity. Hence, each $\mathcal{C}_j$ corresponds to one camera pose hypothesis $\mathbf{h}_j$, which we recover using a pose solver g, i.e.

$$\mathbf{h}_j = g(\mathcal{C}_j). \tag{3}$$

The concrete manifestation of $g(\cdot)$ differs for RGB and RGB-D inputs. Note that the RANSAC algorithm [53] includes a way to adaptively choose the number of hypotheses M according to an online estimate of the outlier ratio in $\mathcal{C}$, i.e. the amount of erroneous correspondences. In this work, and our previous work [5], [14], [15], [16], [17], we choose a fixed M and train the system to adapt to this particular setting. Thereby, M becomes a hyper-parameter that controls the allowance of the neural network f to make inaccurate predictions.

2) Choose Best Hypothesis. Following RANSAC, we choose the hypothesis $\mathbf{h}_j$ with maximum consensus among all scene coordinates $\mathcal{Y}$, i.e.

$$\tilde{\mathbf{h}} = \underset{\mathbf{h}_j}{\operatorname{argmax}}\, s(\mathbf{h}_j, \mathcal{Y}). \tag{4}$$

We measure consensus by a scoring function $s(\cdot)$ that is, by default, implemented as inlier counting:

$$s(\mathbf{h}, \mathcal{Y}) = \sum_{\mathbf{y}_i \in \mathcal{Y}} \mathbb{1}[r(\mathbf{y}_i, \mathbf{h}) < \tau]. \tag{5}$$

Function $r(\cdot)$ measures the residual between pose parameters $\mathbf{h}$, and a scene coordinate $\mathbf{y}_i$, $\mathbb{1}[\cdot]$ evaluates to one if the residual is smaller than an inlier threshold τ.

3) Refine Best Hypothesis. We refine the chosen hypothesis $\tilde{\mathbf{h}}$, which was created from a small subset of correspondences, using all scene coordinates:

$$\hat{\mathbf{h}} = \mathbf{R}(\tilde{\mathbf{h}}, \mathcal{Y}). \tag{6}$$

We implement refinement as re-solving for the pose parameters using the complete inlier set $\mathcal{I}$ of hypothesis $\tilde{\mathbf{h}}$, i.e.

$$\mathbf{R}(\tilde{\mathbf{h}}, \mathcal{Y}) = g(\mathcal{C}_\mathcal{I}) \text{ with } \mathcal{I} = \{i | r(\mathbf{y}_i, \tilde{\mathbf{h}}) < \tau\} \tag{7}$$

We iterate refinement and re-calculation of the inlier set $\mathcal{I}$ until the inlier count stops increasing. We refer to the refined hypothesis as our final camera pose estimate $\hat{\mathbf{h}}$.

Next, we discuss particular choices for pose optimization components in case the input image is RGB or RGB-D .

3.1 Case RGB

In case the input is an RGB image without a depth channel, correspondences $\mathcal{C}$ manifest as 2D-3D correspondences between the image and 3D scene space:

$$\mathcal{C}^{\text{RGB}} = \{(\mathbf{p}_i, \mathbf{y}_i) | \mathbf{y}_i \in \mathcal{Y}\}, \tag{8}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

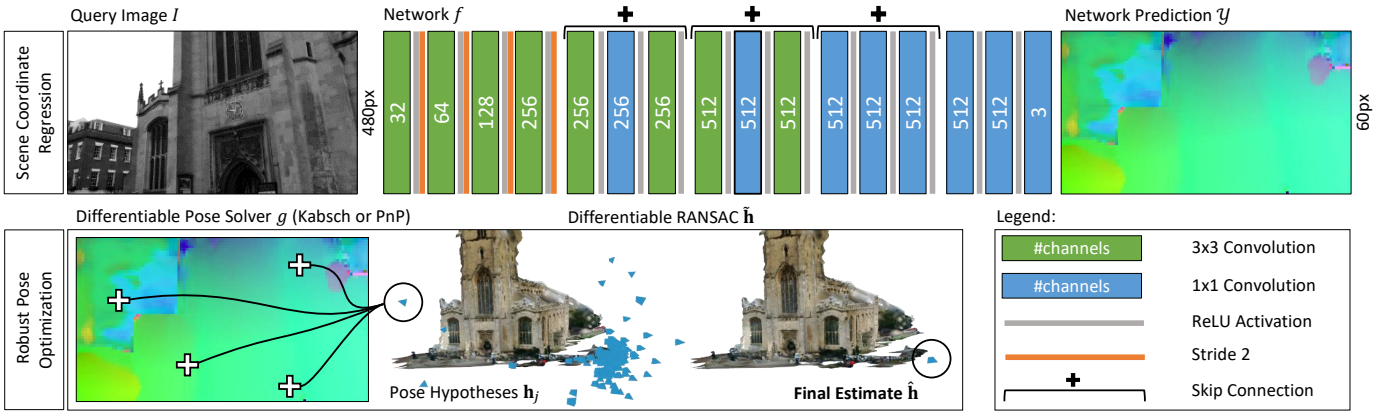


Fig. 3. **System Overview.** The system consists of two stages: Scene coordinate regression using a CNN (**top**) and differentiable pose estimation (**bottom**). The network is fully convolutional and produces a dense but sub-sampled output. Pose estimation employs a minimal solver (PnP [66] for RGB images or Kabsch [49] for RGB-D images) within a RANSAC [53] robust estimator. The final camera pose estimate is also refined. To allow for end-to-end training, all components need to be differentiable. While the Kabsch solver is inherently differentiable, we describe differentiable approximations for PnP and RANSAC.

where $\mathbf{p}_i$ denotes the 2D image coordinate associated with pixel i. Image coordinates and scene coordinates are related by

$$\mathbf{p}_i = K\mathbf{h}^{-1}\mathbf{y}_i \tag{9}$$

where K denotes the camera calibration matrix, or internal calibration parameters of the camera. Using this relation, perspective-n-point (PnP) solvers $g(\cdot)$ [66], [67] recover the camera pose from at least four 2D-3D correspondences for a unique solution: $|\mathcal{C}_j^{\text{RGB}}| \geq 4$. In practise, we use $|\mathcal{C}_j^{\text{RGB}}| = 4$ with the solver of Gao et al. [66] when sampling pose hypotheses $\mathbf{h}_j$, and non-linear optimization of the re-projection error with Levenberg-Marquardt [68], [69] when refining the chosen hypothesis $\mathbf{R}(\tilde{\mathbf{h}}, \mathcal{Y})$ with $|\mathcal{C}_\mathcal{I}| > 4$. We utilize the implementation available in OpenCV [70] for all PnP solvers.

As residual function $r(\cdot)$ for determining the score $s(\cdot)$ of a pose hypothesis $\mathbf{h}_j$ in Eq. 5, we calculate the re-projection error:

$$r^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}) = ||\mathbf{p}_i - K\mathbf{h}^{-1}\mathbf{y}_i||. \tag{10}$$

3.2 Case RGB-D

In case the input is an RGB-D image, the known depth map allows us to recover the 3D coordinate corresponding to each pixel i in the coordinate frame of the camera, denoted as $\mathbf{e}_i$. Together with the scene coordinate prediction $\mathcal{Y}$, we have dense 3D-3D correspondences $\mathcal{C}$ between camera space and scene space, i.e.

$$\mathcal{C}^{\text{RGB-D}} = \{(\mathbf{e}_i, \mathbf{y}_i) | \mathbf{y}_i \in \mathcal{Y}\}. \tag{11}$$

To recover the camera pose from 3D-3D correspondences we utilize the Kabsch algorithm [49], sometimes also called orthogonal Procrustes, as pose solver $g(\cdot)$. For sampling pose hypotheses $\mathbf{h}_j$, we use $|\mathcal{C}_j^{\text{RGB-D}}| = 3$, when refining the chosen hypothesis $\mathbf{R}(\tilde{\mathbf{h}}, \mathcal{Y})$ we use $|\mathcal{C}_\mathcal{I}| > 3$.

As residual function $r(\cdot)$ for determining the score $s(\cdot)$ of an hypothesis $\mathbf{h}_j$ in Eq. 5, we calculate the 3D Euclidean distance:

$$r^{\text{RGB-D}}(\mathbf{y}_i, \mathbf{h}) = ||\mathbf{e}_i - \mathbf{h}^{-1}\mathbf{y}_i||. \tag{12}$$

4 Deep Scene Coordinate Regression

In this section, we discuss the neural network architecture for scene coordinate regression, and how to train it using auxiliary losses defined on the scene coordinate output. These auxiliary losses serve as an initialization step prior to training the whole pipeline in an end-to-end fashion, see Sec. 5. The initialization is necessary, since end-to-end training from scratch will converge to a local minimum without giving reasonable pose estimates.

We implement scene coordinate regression $f(\cdot)$ using a fully convolutional neural network [13] with skip connections [18] and learnable parameters $\mathbf{w}$. We depict the network architecture in Fig. 3, top. The network takes a single channel grayscale image as input, and produces a dense scene coordinate prediction sub-sampled by the factor 8. Sub-sampling, implemented with stride 2 convolutions, increases the receptive field associated with each pixel output while also enhancing efficiency. The total receptive field of each output scene coordinate is 81px. In experiments on various datasets, we found no advantage in providing the full RGB image as input, in contrast, conversion to grayscale slightly increases the robustness to non-linear lighting effects.

Relation to our Previous Work. In our first DSAC-based re-localization pipeline [14] and in DSAC++ [15], we utilized a VGGNet-style architecture [71]. It had a larger memory footprint and slower runtime while offering similar accuracy. The receptive field was comparable with 79px. In the experiments of Sec. 6, we conduct an empirical comparison of both architectures. We utilized our updated architecture already in our work on ESAC [17] and NG-RANSAC [16].

In the following, we discuss different strategies on initializing the scene coordinate neural network, depending on what information is available for training. In particular, we discuss training from RGB-D images for RGB-D-based re-localization, training from RGB images and a 3D model of the scene for RGB-based re-localization as well as training from RGB images only for RGB-based re-localization. See Table 1 for a schematic overview. Other combinations are of course possible, e.g. training from RGB images only, but

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 1
Information Available at Training and Test Time. "D" stands for depth channel, "poses" stands for ground truth camera poses. The 3D model of the scene may be a sparse point cloud, e.g. from a SfM reconstruction [9], [10], or a dense 3D scan [72], [73], [74].

	Training				Test	
Setting	RGB	D	poses	3D model	RGB	D
RGB-D	✓	✓	✓		✓	✓
RGB + 3D model	✓		✓	✓	✓	
RGB	✓		✓		✓	

having RGB-D images at test time. However, we restrict our discussion and experiments to the most common settings found in the literature [3], [5], [15].

4.1 RGB-D

For RGB-D-based pose estimation, we initialize our neural network by minimizing the Euclidean distance between predicted scene coordinates $\mathbf{y}_i$ and ground truth scene coordinates $\mathbf{y}_i^*$.

$$\ell^{\text{RGB-D}}(\mathbf{y}_i, \mathbf{y}_i^*) = ||\mathbf{y}_i^* - \mathbf{y}_i|| \tag{13}$$

We obtain ground truth scene coordinates $\mathbf{y}_i^*$ by re-projecting depth channels of training images to obtain 3D points $\mathbf{e}_i$ in the camera coordinate frame, and transforming them using the ground truth pose $\mathbf{h}^*$, i.e. $\mathbf{y}_i^* = \mathbf{h}^*\mathbf{e}_i$. We train the network using the average loss over all pixels of a training image:

$$\mathcal{L}^{\text{RGB-D}}(\mathcal{Y}, \mathcal{Y}^*) = \frac{1}{|\mathcal{Y}|} \sum_{\mathbf{y}_i \in \mathcal{Y}} \ell^{\text{RGB-D}}(\mathbf{y}_i, \mathbf{y}_i^*). \tag{14}$$

We motivate optimizing the Euclidean distance for RGB-D-based re-localization by the fact that the corresponding Kabsch pose solver optimizes the pose over squared Euclidean residuals between camera coordinates and scene coordinates. We found the plain, instead of the squared, Euclidean distance in Eq. 13 superior in [14] due to its robustness to outliers.

4.2 RGB + 3D Model

In case the camera pose is to be estimated from an RGB image, the optimization of scene coordinates w.r.t. a 3D Euclidean distance is not optimal. The PnP solver, which we utilize for pose sampling and pose refinement, optimizes the camera pose w.r.t. the re-projection error of scene coordinates. Hence, for RGB-based pose estimation, we initialize the scene coordinate regression network by minimizing the re-projection error of its predictions, i.e. $r^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*)$ where $r^{\text{RGB}}(\cdot)$ denotes the residual function defined for RGB in Eq. 10, and $\mathbf{h}^*$ denotes the ground truth camera pose.

Unfortunately, optimizing this objective from scratch fails since the re-projection error is ambiguous w.r.t. the viewing direction of the camera. However, if we assume a 3D model of the environment to be available, we may render ground truth scene coordinates $\mathcal{Y}^*$, optimize the RGB-D objective of Eq. 13 first, and switch to the re-projection error after a few training iterations:

$$\ell^{\text{RGB+M}}(\mathbf{y}_i, \mathbf{y}_i^*, \mathbf{h}^*) = \begin{cases} \hat{r}^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*) & \text{if } \mathbf{y}_i \in \mathcal{V}^{\text{RGB+M}} \\ ||\mathbf{y}_i^* - \mathbf{y}_i|| & \text{otherwise.} \end{cases} \tag{15}$$

We define a set of valid scene coordinate predictions as $\mathcal{V}^{\text{RGB+M}}$ for which we optimize the re-projection error. If a scene coordinate does not qualify as valid yet, we optimize the Euclidean distance, instead. A prediction $\mathbf{y}_i$ is valid, $\mathbf{y}_i \in \mathcal{V}^{\text{RGB+M}}$ if:

1) $(\mathbf{h}^{*-1}\mathbf{y}_i)_z > 0.1$m, i.e. it lies at least 0.1m in front of the ground truth image plane.
2) It has a maximum re-projection error of $r^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*) < 1000$px.
3) It is within a maximum 3D distance w.r.t. to the rendered ground truth coordinate of $||\mathbf{y}_i^* - \mathbf{y}_i|| < 0.1$m.

The training objective is flexible w.r.t. to missing ground truth scene coordinates for certain pixels, i.e. if $\mathbf{y}_i^* = \mathbf{0}$. In this case, we only enforce constraint 1) and 2) for $\mathcal{V}^{\text{RGB+M}}$. This allows us to utilize dense 3D models of the scene, sparse SfM reconstructions as well as depth channels with missing measurements to generate $\mathbf{y}_i^*$. The training objective utilizes a robust version $\hat{r}^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*)$ of the RGB residual function of Eq. 10, i.e.

$$\hat{r}^{\text{RGB}}(\mathbf{y}, \mathbf{h}) = \begin{cases} r^{\text{RGB}}(\mathbf{y}, \mathbf{h}) & \text{if } r^{\text{RGB}}(\mathbf{y}, \mathbf{h}) < 100\text{px} \\ \sqrt{100 r^{\text{RGB}}(\mathbf{y}, \mathbf{h})} & \text{otherwise.} \end{cases} \tag{16}$$

This formulation implements a soft clamping by using the square root of the re-projection residual after a threshold of 100px. To train the scene coordinate network, we optimize the average of Eq. 15 over all pixels of a training image, similar to Eq. 14.

Relation to our Previous Work. We introduced a combined training objective based on, firstly, minimizing the 3D distance to ground truth scene coordinates, and, secondly, minimizing the re-projection error in DSAC++ [15]. However, DSAC++ uses separate initalization stages for the two objectives, 3D distance and re-projection error, which is computationally wasteful. The network might concentrate on modelling fine details of the geometry in the first initialization stage which is potentially undone in the second initialization stage. Also, pixels without a ground truth scene coordinate would receive no training signal in the first initalization stage of DSAC++. The new, combined training objective of DSAC* in Eq. 15 switches dynamically from optimizing the 3D distance to optimizing the re-projection error on a per-pixel basis. By using one combined initialization stage instead of two, we cut the pre-training time of DSAC* in half compared to DSAC++ on identical hardware.

4.3 RGB

The previous RGB-based training objective of Eq. 15 relies on the availability of a 3D model of the scene. When a dense 3D scan of an environment is unavailable, SfM tools like VisualSfM [9] or COLMAP [10] offer workable solutions to create a (sparse) 3D model from a collection of RGB images, e.g. from the training set of a scene. However, for some environments, particularly indoors, a SfM reconstruction might fail due to texture-less areas or repeating structures. Also, despite SfM tools having matured significantly over many years since the introduction of Bundler [8] they still represent expert tools with their own set of hyper-parameters to be tuned. Therefore, it might be attractive to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

train a camera re-localization system from RGB images and ground truth poses alone, without resorting to an SfM tool for pre-processing. Therefore, we introduce a variation on the RGB-based training objective of Eq. 15 that substitutes ground truth scene coordinates $\mathbf{y}_i^*$ with a heuristic scene coordinate target $\bar{\mathbf{y}}_i$ combined with a robust L_1 distance:

$$\ell^{\text{RGB}}(\mathbf{y}_i, \bar{\mathbf{y}}_i, \mathbf{h}^*) = \begin{cases} \hat{r}^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*) & \text{if } \mathbf{y}_i \in \mathcal{V}^{\text{RGB}} \\ |\bar{\mathbf{y}}_i - \mathbf{y}_i| & \text{otherwise.} \end{cases} \tag{17}$$

We obtain heuristic targets $\bar{\mathbf{y}}_i = \mathbf{h}^* \bar{\mathbf{e}}_i$ from the ground truth camera pose $\mathbf{h}^*$ and dummy 3D camera coordinates $\bar{\mathbf{e}}_i$ re-projected while assuming a constant image depth of 10m. The above formulation relies on switching from the heuristic target to the re-projection error as soon as possible. Therefore, we formulate the following relaxed validity constraints for scene coordinate predictions $\mathbf{y}_i$ to form the set $\mathcal{V}^{\text{RGB}}$:

1) $({\mathbf{h}^*}^{-1}\mathbf{y}_i)_z > 0.1$m, i.e. it lies at least 0.1m in front of the ground truth image plane.
2) $({\mathbf{h}^*}^{-1}\mathbf{y}_i)_z < 1000$m, i.e. it lies at most 1000m in front of the ground truth image plane.
3) It has a maximum re-projection error of $r^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}^*) < 1000$px.

When we were able to constrain scene coordinate predictions to be near the ground truth geometry in Sec. 4.2, here we merely enforce a coarse range of plausibility via item 2.

Relation to our Previous Work. DSAC++ [15] used two separate initialization stages for minimizing the distance to heuristic targets $\bar{\mathbf{y}}_i$, and optimization of the re-projection error, respectively. The first initialization stage was particularly cumbersome since the heuristic targets $\bar{\mathbf{y}}_i$ are inconsistent w.r.t. the true 3D geometry of the scene. The neural network can easily overfit to $\bar{\mathbf{y}}_i$ which we circumvent in DSAC++ by early stopping and by using only a fraction of the full training data for the first initialization stage. The new, combined formulation of Eq. 17 is more robust by only loosely enforcing the heuristic until the formulation adaptively switches to the re-projection error. Also, as mentioned in the previous section, the new formulation is more efficient by combining two initialization stages into one, thus reducing training time.

5 Differentiable Pose Optimization

Our overall goal is training the complete pose estimation pipeline in an end-to-end fashion. That is, we wish to optimize the learnable parameters $\mathbf{w}$ of scene coordinate prediction in a way that we obtain highly accuracy pose estimates $\hat{\mathbf{h}}$ as per Eq. 6 and Eq. 4. Due to the robust nature of our pose optimization, particularly due to deploying RANSAC to estimate model parameters, the relation of the quality of scene coordinates $\mathcal{Y}$ and the estimated pose $\hat{\mathbf{h}}$ is non-trivial. For example, any prediction $\mathbf{y}_i$ that RANSAC labels as an outlier will not influence $\hat{\mathbf{h}}$ at all. Improving the accuracy of outlier scene coordinates will not help us in obtaining higher pose quality. To the contrary, it might be beneficial to *decrease* the accuracy of outlier scene coordinates further to make sure that RANSAC recognises them as outliers. However, we have no prior knowledge which exact predictions for an image should be inliers or outliers of the estimated model.

In this work, we address this problem by making pose optimization itself differentiable, to include it in the training process. By training in an end-to-end fashion, the scene coordinate network may adjust its predictions in any way that results in accurate pose estimates. More formally, we define the following loss function on estimated poses:

$$\ell^{\text{Pose}}(\hat{\mathbf{h}}, \mathbf{h}^*) = ||\hat{\mathbf{t}} - \mathbf{t}^*|| + \gamma \measuredangle(\hat{\boldsymbol{\theta}}, \boldsymbol{\theta}^*), \tag{18}$$

with $\mathbf{h} = (\boldsymbol{\theta}, \mathbf{t})$ consisting of translation parameters $\mathbf{t}$ and rotation parameters $\boldsymbol{\theta}$. We denote the ground truth pose parameters as $\mathbf{t}^*$ and $\boldsymbol{\theta}^*$ respectively. The weighting factor γ controls the trade-off between translation and rotation accuracy. We use $\gamma = 100$ in our work, comparing rotation in degree to translation in cm. Similar to Eq. 16, we robustify the pose loss by soft clamping.

$$\hat{\ell}^{\text{Pose}}(\hat{\mathbf{h}}, \mathbf{h}^*) = \begin{cases} \ell^{\text{Pose}}(\hat{\mathbf{h}}, \mathbf{h}^*) & \text{if } \ell^{\text{Pose}}(\hat{\mathbf{h}}, \mathbf{h}^*) < 100 \\ \sqrt{100 \ell^{\text{Pose}}(\hat{\mathbf{h}}, \mathbf{h}^*)} & \text{otherwise.} \end{cases} \tag{19}$$

The estimated camera pose $\hat{\mathbf{h}}$ depends on network parameters $\mathbf{w}$ via the network prediction $\mathcal{Y}$ through robust pose optimization. In order to optimize the pose loss of Eq. 19, each component involved in pose optimization needs to be differentiable. In the remainder of this section, we discuss the differentiability of each component and derive approximate gradients where necessary. We discuss the differentiability of the Kabsch [49] pose solver for RGB-D images in Sec. 5.1. We give an analytical approximation for gradients of PnP solvers for RGB-based pose estimation in Sec. 5.2. In Sec. 5.3, we explain how to approximate gradients of iterative pose refinement. We discuss differentiable pose scoring via soft inlier counting in Sec. 5.4. Finally, we present a differentiable version of RANSAC, called *differentiable sample consensus* (DSAC) in Sec. 5.5 which also defines our overall training objective.

5.1 Differentiating Kabsch

We utilize the Kabsch pose solver when estimating poses from RGB-D inputs. In this setting, we have 3D-3D correspondences $\mathcal{C}^{\text{RGB-D}}(\mathcal{Y})$ given between the 3D coordinates in camera space, defined by the given depth map, and 3D coordinates in scene space $\mathcal{Y}$ predicted by our neural network. In the following, we assume that we apply the Kabsch solver $g^{\text{Kabsch}}(\cdot)$ over a subset of correspondences $\mathcal{C}_{\mathcal{I}}(\mathcal{Y})$ either when sampling pose hypotheses from three correspondences, or refining the final pose estimate over an inlier set found by RANSAC:

$$\mathbf{h}(\mathcal{Y}) = g^{\text{Kabsch}}(\mathcal{C}_{\mathcal{I}}(\mathcal{Y})) \text{ with } \mathcal{C}_{\mathcal{I}}(\mathcal{Y}) \subseteq \mathcal{C}^{\text{RGB-D}}(\mathcal{Y}) \tag{20}$$

Here, and in the following, we make the dependence of a model hypothesis to the scene coordinate prediction explicit, i.e. we write $\mathbf{h}(\mathcal{Y})$. The Kabsch solver returns the pose that minimizes the squared residuals over all correspondences:

$$g^{\text{Kabsch}}(\mathcal{C}_{\mathcal{I}}(\mathcal{Y})) = \underset{\mathbf{h}'}{\text{argmin}} \sum_{(\mathbf{e}_i, \mathbf{y}_i) \in \mathcal{C}_{\mathcal{I}}(\mathcal{Y})} r^{\text{RGB-D}}(\mathbf{y}_i, \mathbf{h}')^2. \tag{21}$$

The optimization can be solved in closed form by the following steps [49]. Firstly, we calculate the covariance matrix $\text{cov}[\cdot]$ over the correspondence set:

$$\text{cov}[\mathcal{C}_{\mathcal{I}}(\mathcal{Y})] = \sum_{(\mathbf{e}_i, \mathbf{y}_i) \in \mathcal{C}_{\mathcal{I}}(\mathcal{Y})} (\mathbf{e}_i - \bar{\mathbf{e}})(\mathbf{y}_i - \bar{\mathbf{y}})^T, \tag{22}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

where $\bar{\mathbf{e}}$ and $\bar{\mathbf{y}}$ denote the mean over all 3D coordinates in the correspondence set in camera space and scene space, respectively. Secondly, we apply a singular value decomposition (SVD) to the covariance matrix:

$$\text{cov}[\mathcal{C}_\mathcal{I}(\mathcal{Y})] = U\Sigma V^T. \tag{23}$$

We re-assemble the optimal rotation $\boldsymbol{\theta}$, and, subsequently, recover the optimal translation $\mathbf{t}$:

$$\begin{aligned} \boldsymbol{\theta} &= V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix} U^T \\ \mathbf{t} &= \bar{\mathbf{y}} - \boldsymbol{\theta}(\bar{\mathbf{e}}) \\ \mathbf{h}(\mathcal{Y}) &= (\boldsymbol{\theta}, \mathbf{t}). \end{aligned} \tag{24}$$

All operations involved in the calculation of $g^{\text{Kabsch}}(\cdot)$ are differentiable, particularly the gradients of SVD can be calculated according to [75], with current deep learning frameworks like PyTorch [19] offering corresponding implementations. The differentiability of the Kabsch algorithm has e.g. also been utilized in [76].

5.2 Differentiable PnP

Similar to the Kabsch solver of the previous section, the PnP solver $g^{\text{PnP}}(\cdot)$ calculates a pose estimate over a subset $\mathcal{C}_\mathcal{I}(\mathcal{Y})$ of all correspondences $\mathcal{C}^{\text{RGB}}(\mathcal{Y})$, i.e.

$$\mathbf{h}(\mathcal{Y}) = g^{\text{PnP}}(\mathcal{C}_\mathcal{I}(\mathcal{Y})) \text{ with } \mathcal{C}_\mathcal{I}(\mathcal{Y}) \subseteq \mathcal{C}^{\text{RGB}}(\mathcal{Y}). \tag{25}$$

We utilize a PnP solver when estimating camera poses from RGB images, where 2D-3D correspondences are given between 2D image positions $\mathbf{p}_i$ and 3D scene coordinate $\mathbf{y}_i \in \mathcal{Y}$. A PnP solver optimizes pose parameters to minimize squared re-projection errors:

$$g^{\text{PnP}}(\mathcal{C}_\mathcal{I}(\mathcal{Y})) = \underset{\mathbf{h}'}{\text{argmin}} \, ||\mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h}')||^2. \tag{26}$$

We construct a residual vector $\mathbf{r}_\mathcal{I}(\cdot)$ over all pixels associated with the current correspondence subset:

$$\mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h})_i = \begin{cases} r^{\text{RGB}}(\mathbf{y}_i, \mathbf{h}) & \text{if } (\mathbf{p}_i, \mathbf{y}_i) \in \mathcal{C}_\mathcal{I}(\mathcal{Y}) \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

where $r^{\text{RGB}}(\cdot)$ denotes a pixels re-projection error, cf. Eq. 10.

In contrast to the Kabsch optimization objective, we cannot solve the PnP objective of Eq. 26 in closed form. Different PnP solvers have been proposed in the past with different algorithmic structures, e.g. [66], [77] or the Levenberg-Marquardt-based optimization in OpenCV [68], [69], [70]. Instead of trying to propose a differentiable variant of the aforementioned PnP algorithms, we calculate an analytical approximation of PnP gradients derived from the objective function in Eq. 26 [78]. We have introduced this way of differentiating PnP in the context of neural network training in DSAC++ [15].

Given a proper initialization, e.g. by [66], [77], we can optimize Eq. 26 iteratively using the Gauss-Newton method. Since we are interested only in the gradients of the optimal pose parameters found at the end of optimization, we ignore the influence of initialization itself, avoiding to calculate gradients of complex minimal solvers like [66], [77]. We give the Gauss-Newton update step to model parameters as

$$\mathbf{h}^{t+1} = \mathbf{h}^t - J_\mathbf{r}^+ \mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h}^t), \tag{28}$$

where $J_\mathbf{r}^+ = (J_\mathbf{r}^T J_\mathbf{r})^{-1} J_\mathbf{r}^T$ is the pseudoinverse of the Jacobian matrix $J_\mathbf{r}$ of the residual vector $\mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h})$ defined in Eq. 27. In particular, the Jacobian matrix $J_\mathbf{r}$ is comprised of the following partial derivatives:

$$(J_\mathbf{r})_{ij} = \frac{\partial \mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h}^t)_i}{\partial \mathbf{h}_j^t}. \tag{29}$$

As mentioned before, the initial pose $\mathbf{h}^{t=0}$ may be provided by an arbitrary, non-differentiable PnP algorithm [66], [77]. We define the pose estimate of the PnP solver as the pose parameters after convergence of the associated optimization problem.

$$\mathbf{h}(\mathcal{Y}) = g^{\text{PnP}}(\mathcal{C}_\mathcal{I}(\mathcal{Y})) = \mathbf{h}^{t=\infty} \tag{30}$$

Thus, we may calculate approximate gradients of model parameters $\mathbf{h}(\mathcal{Y})$ w.r.t. scene coordinates $\mathcal{Y}$ by fixing the last optimization iteration around the final model parameters:

$$\frac{\partial}{\partial \mathcal{Y}} \mathbf{h}(\mathcal{Y}) \approx -J_\mathbf{r}^+ \frac{\partial}{\partial \mathcal{Y}} \mathbf{r}_\mathcal{I}(\mathcal{Y}, \mathbf{h}^{t=\infty}). \tag{31}$$

5.3 Differentiable Refinement

We refine given camera pose parameters $\mathbf{h}$, denoted as $\mathbf{R}(\mathbf{h}, \mathcal{Y})$, by iteratively re-solving for the pose using the set of all inliers $\mathcal{I}$, and updating the set of inliers with the new pose estimate:

$$\begin{aligned} \mathbf{h}^{t+1} &= g(\mathcal{C}_{\mathcal{I}^t}) \\ \mathcal{I}^{t+1} &= \{i | r(\mathbf{y}_i, \mathbf{h}^{t+1}) < \tau\} \end{aligned} \tag{32}$$

We repeat refinement until the inlier set $\mathcal{I}$ ceases to grow, i.e. $\mathbf{R}(\mathbf{h}, \mathcal{Y}) = g(\mathcal{C}_{\mathcal{I}^{t=\infty}})$ where $\mathcal{I}^{t=\infty}$ corresponds to the final inlier set. Similar to differentiating PnP in the previous section, we approximate gradients of iterative refinement by fixing the last refinement iteration.

$$\frac{\partial}{\partial \mathcal{Y}} \mathbf{R}(\mathbf{h}, \mathcal{Y}) = \frac{\partial}{\partial \mathcal{Y}} g(\mathcal{C}_{\mathcal{I}^{t=\infty}}), \tag{33}$$

where function $g(\cdot)$ denotes either the Kabsch solver or the PnP solver for RGB-D and RGB inputs, respectively. We have discussed the calculation of gradients for $g(\cdot)$ already in the previous sections.

5.4 Differentiable Inlier Count

We obtain a differentiable approximation of inlier counting of Eq. 5 by substituting the hard comparison of a pixel's residual to an inlier threshold τ with a Sigmoid function $\sigma[\cdot]$:

$$s(\mathbf{h}, \mathcal{Y}) = \sum_{\mathbf{y}_i \in \mathcal{Y}} \sigma[\beta\tau - \beta r(\mathbf{y}_i, \mathbf{h})]. \tag{34}$$

For hyper-parameter β, which controls the softness of the Sigmoid function, we use the following heuristic in dependence of the inlier threshold τ: $\beta = \frac{5}{\tau}$.

Relation to our Previous Work. In the original DSAC pipeline [14] we utilize a designated scoring CNN as a differentiable alternative to traditional inlier counting. However, our follow-up work on DSAC++ [15] revealed that a scoring CNN is prone to overfitting, and does in general not exceed the accuracy of the simpler soft inlier count.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

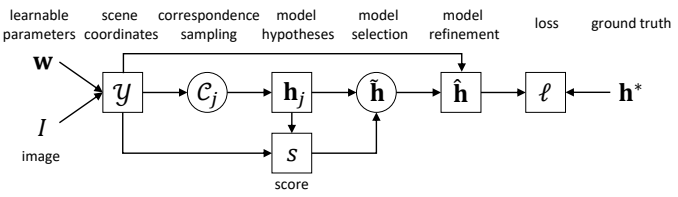


Fig. 4. **DSAC Computation Graph [79].** Nodes without frames represent inputs to the system, nodes with square frames represent deterministic operations, nodes with circular frames represent sampling operations. Arrows denote an input relation.

5.5 Differentiable RANSAC

We can subsume the RANSAC algorithm [53] in the following three steps, as also discussed in Sec. 3: Firstly, generate model hypotheses by random sampling of correspondences. Secondly, choose the best hypothesis according to a scoring function. Lastly, refine the winning hypotheses using its inliers. We discussed the differentiability of most components involved in the previous sub-sections, e.g. calculating gradients of pose solvers used for hypothesis sampling and refinement, and differentiating the inlier count for hypothesis scoring. However, choosing the best hypothesis according to Eq. 4 involves a non-differentiable argmax operation.

In [14], we introduce a differentiable approximation of hypothesis selection in RANSAC, called *differentiable sample consensus* (DSAC). In [14], we also argue, and show empirically, that a simple soft argmax approximation, a weighted average of arguments, does not work well. A soft argmax can be unstable when arguments have multi-modal structure, i.e. very different arguments have high weights in the average. A standard average might also be overly sensitive to outlier arguments.

The DSAC approximation relies on a probabilistic selection of a model hypothesis according to a probability distribution $p(j|\mathcal{Y})$ over the discrete set of sampled hypotheses with index j:

$$\begin{aligned} \tilde{\mathbf{h}}(\mathcal{Y}) &= \mathbf{h}_j(\mathcal{Y}) \text{ with } j \sim p(j|\mathcal{Y}) \\ \hat{\mathbf{h}}(\mathcal{Y}) &= \mathbf{R}(\tilde{\mathbf{h}}(\mathcal{Y}), \mathcal{Y}), \end{aligned} \tag{35}$$

where $\tilde{\mathbf{h}}(\mathcal{Y})$ denotes the selected hypothesis, and $\hat{\mathbf{h}}(\mathcal{Y})$ denotes the final, refined estimate of our pipeline. The distribution guiding hypothesis selection is a softmax distribution over scores, i.e.

$$p(j|\mathcal{Y}) = \frac{\exp[\alpha s(\mathbf{h}_j(\mathcal{Y}), \mathcal{Y})]}{\sum_{k=1}^{M} \exp[\alpha s(\mathbf{h}_k(\mathcal{Y}), \mathcal{Y})]}. \tag{36}$$

The hyper-parameter α corresponds to a temperature that controls the softness of the distribution. The larger α, the more DSAC will behave like RANSAC in always selecting the hypothesis with maximum score, while providing less signal for learning. In DSAC++ [15], we present a schema to adjust α automatically during learning. In this work, we treat α as a hand-tuned and fixed hyper-parameter, as we found the camera re-localization problem not overly sensitive to the exact value of α, and fixing it simplifies the software architecture of our pipeline.

To learn the pipeline, we optimize the expectation of the pose loss $\hat{\ell}^{\text{Pose}}$ of Eq. 19 w.r.t. randomly selecting hypotheses:

$$\mathcal{L}^{\text{Pose}}(\mathcal{Y}, \mathbf{h}^*) = \mathbb{E}_{j \sim p(j|\mathcal{Y})} \left[\hat{\ell}^{\text{Pose}}(\mathbf{R}(\cdot), \mathbf{h}^*) \right], \tag{37}$$

where we abbreviate the final, refined camera pose $\mathbf{R}(\mathbf{h}_j(\mathcal{Y}), \mathcal{Y})$ as $\mathbf{R}(\cdot)$. To minimize the expectation, the neural network should learn to predict scene coordinates $\mathcal{Y}$ that ensure the following two properties: Firstly, hypotheses with a large loss after refinement should receive a low selection probability, i.e. a low soft inlier count. Secondly, hypotheses with a high soft inlier count should receive a small loss after refinement. We present a schematic overview of all components involved in our DSAC-based pipeline in Fig. 4. The figure summarises dependencies between processing steps, and differentiates between deterministic functions and sampling operations. The graph structure illustrates the non-trivial relation between the scene coordinate prediction and pose quality, since scene coordinates directly influence pose hypotheses, scoring and refinement.

The DSAC training objective of Eq. 37 is smooth and differentiable, and its gradients can be formulated as follows:

$$\frac{\partial}{\partial \mathcal{Y}} \mathcal{L}^{\text{Pose}}(\cdot) = \mathbb{E}_j \left[\hat{\ell}^{\text{Pose}}(\cdot) \frac{\partial}{\partial \mathcal{Y}} \log p(j|\mathcal{Y}) + \frac{\partial}{\partial \mathcal{Y}} \hat{\ell}^{\text{Pose}}(\cdot) \right], \tag{38}$$

where we use $\cdot$ as a stand-in for the respective function arguments in Eq. 37, and abbreviate the expectation over $p(j|\mathcal{Y})$ as $\mathbb{E}_j[\cdot]$. We use Eq. 38 to learn our system in an end-to-end fashion, updating neural network parameters $\mathbf{w}$ of scene coordinate prediction $\mathcal{Y} = f(I, \mathbf{w})$.

6 Experiments

We evaluate our camera re-localization pipeline on two indoor datasets and one outdoor dataset. Firstly, in Sec. 6.1 we discuss our experimental setup, including datasets, training schedule, hyper-parameters and competitors. Secondly, we report results on 3 different datasets in Sections 6.2, 6.3 and 6.4, respectively. Thirdly, we provide several ablation studies in Sections 6.5, 6.6, 6.7 and 6.8, as well as visualizations of scene representations learned by our system in Sec. 6.9. Furthermore, we analyze the scene compression properties of DSAC* in Sec. 6.10.

6.1 Setup

Task Variants. We deploy our system in several flavours, catering to different application scenarios where depth measurements or 3D scans of a scene might be available or not. Specifically, we analyze the following settings:

- **RGB-D:** We have RGB-D images for training as well as at test time. For initialization training, we render ground truth scene coordinates using 3D scans of each scene. For end-to-end training and at test time, we generate camera coordinates $\mathbf{e}$ from the RGB-D depth channels. We use a Kabsch [49] solver for sampling hypotheses and for refining the final estimate.
- **RGB + 3D model:** We have RGB images for training as well as at test time. We can render ground truth scene coordinates for training using a 3D model of the scene. The 3D model can either be a sparse

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

SfM point cloud, or a dense 3D scan. We use the PnP solver of Gao et al. [66] to sample camera pose hypotheses, and the Levenberg-Marquardt [68], [69] PnP optimizer of OpenCV [70] for final refinement.

- **RGB:** Same as the previous setting, but we have no information about the 3D geometry of a scene, only RGB images and ground truth poses for training. To initialize scene coordinate regression, we optimize the heuristic objective of Eq. 13.

Hyper-Parameters. We convert input images to grayscale and re-scale them to 480px height. For training, we follow Li et al. [54] and apply data augmentation. We apply random adjustments of brightness and contrast of the input image within a $\pm 10\%$ range. We randomly rotate images, ground truth scene coordinates and camera poses within a $\pm 30^\circ$ range. We randomly re-scale images within 66% and 150%, and adjust the focal length accordingly. Different from Li et al. [54], we do not shear training images, since our simple pinhole camera model does not support this operation. We also do not shift images. For a patch-based network shifting by more than 4px would just increase the period of the input without any effect other than increasing boundary effects.

We use an inlier threshold $\tau = 10$px for RGB-based pose optimization, and $\tau = 10$cm for RGB-D-based pose optimization. We sample $M = 64$ RANSAC hypotheses. We reject an hypothesis if the corresponding minimal set does not satisfy the inlier threshold [80], and sample again. We score hypotheses using a soft inlier count at training and test time. For training, we optimize the expectation over hypothesis selection according to Eq. 36 with a temperature of $\alpha = \frac{100}{|\mathcal{Y}|}$, where $|\mathcal{Y}|$ corresponds to the number of scene coordinates predicted.At test time, we choose the best hypothesis with highest score like standard RANSAC. We do at most 100 refinement steps, but stop early if the inlier set ceases to grow which typically takes at most 10 steps. We found the pose estimation parameters to be very stable, and use the same values across all scenes, indoor and outdoor.

We initialize the scene coordinate network for 1M iterations, a batch size of 1 image, and the Adam optimizer [81] with a learning rate of 10^{-4}. We train the system end-to-end for another 100k iterations, and a learning rate of 10^{-6}. Training time varies with hardware. The two training stages take ca. 48+12h on a Tesla K80 GPU, or 8+8h on a GeForce RTX 2080 Ti. Pre-training on unrelated data did not speed up training in our experiments. Presumably, the majority of training is needed to encode the scene specific geometry into the network. Our implementation, based on PyTorch [19], is publicly available: https://github.com/vislearn/dsacstar.

Datasets. We evaluate our pipeline on three standard camera re-localization datasets, both indoor and outdoor:

- **7Scenes [3]:** A RGB-D indoor re-localization dataset of seven small indoor environments featuring difficult conditions such as motion blur, reflective surfaces, repeating structures and texture-less areas. Images were recorded using KinectFusion [72] which also provides ground truth camera poses. For each scene, several thousand frames are available which the authors split into training and test sets. The depth channels of this dataset are not registered to the color images. We register them by projecting the depth maps to 3D points using the depth sensor calibration, and re-projecting them using the color sensor calibration while taking the relative transformation between depth and color sensor into account. A dense 3D scan of each scene is available for rendering ground truth coordinates for initialization training.
- **12Scenes [82]:** A RGB-D indoor re-localization dataset similar to 7Scenes, but containing twelve slightly larger indoor environments. Each scene comes with several hundred frames, split by the authors into training and test sets. The depth maps provided by the authors are registered to the color images. A dense 3D scan of each scene is available as well, which we use to render ground truth scene coordinates for initialization training.
- **Cambridge [26]**: A RGB outdoor re-localization dataset of five landmarks in Cambridge, UK. Each landmark spans an area of several hundred or thousand square meters. Each scene comes with several hundred frames, split by the authors into training and test sets. Ground truth camera poses are reconstructed using SfM. We use the sparse SfM point clouds to render sparse scene coordinate ground truth for RGB-based re-localization. The dataset contains a sixth scene, an entire street scene. The corresponding reconstruction is of low quality containing outlier camera poses and 3D points as well as duplicated geometry. These defects prevent convergence of training as we have observed in previous work [15], [16] and as also reported in [11], [28]. Since the problems are with the ground truth itself, interpreting results relative to this ground truth is difficult. Therefore, we omit the street scene in our discussion.

Competitors. We compare to the following **absolute pose regression** networks: PoseNet (the updated version of 2017) [27], SpatialLSTM [11], MapNet [29] and SVS-Pose [28]. We compare to the following **relative pose estimation** approaches: AnchorNet [31], and retrieval-based InLoc [25]. For feature-based competitors, we report results of the ORB baseline used in [3] and [82], as well as the SIFT baseline used in [82]. For a state-of-the-art feature-based pipeline, we compare to ActiveSearch [7]. Several early scene coordinate regression works were based on **random forests**. We compare to SCoRF of Shotton et al. [3], and its extension to multi-output forests (*MO Forests*) [47] and forests predicting Gaussian mixture models (GMM) of scene coordinates, in the variation of Valentain et al. [4] for RGB-D (*GMM F. (V)*) and of Brachmann et al. [5] for RGB (*GMM F. (B)*). Furthermore, we compare to the Back-Tracking Forests of Meng et al. [50] (*BTBRF*), to the Point-Line Forests of Meng et al. [48] (*PLForests*), and MNG forests [82]. For **CNN-based** scene coordinate regression, we compare to ForestNet [51], scene coordinate regression with an angle-based loss [52] (*ABRLoss*), joint scene coordinate classification and regression [54] (*SCoCR*) and the visual descriptor learning approach of Schmidt et al. [6] (*SS-VDL*). We also compare to the *headline performance* of the adaptive forests of Cavallari et al. [56] (*OtF Forests*). Finally, we compare to previous iterations of this pipeline, namely DSAC [14] and DSAC++ [15]. We denote our updated pipeline as DSAC*.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

6.2 Results for Indoor Localization (7Scenes)

We train one scene coordinate regression network per scene, and accept a pose estimate for a test image if its pose error is below 5° and 5cm. We calculate the accuracy per scene, and report the average accuracy over all 7Scenes, see quantitative results in Fig. 5, left.

RGB. For training from RGB images and ground truth poses only, our new training procedure and network architecture increases accuracy significantly compared to DSAC++ (+27.6%). DSAC* also achieves higher accuracy than the angle-based loss of Li et al. [52], despite the latter incorporating multi-view constraints and a photometric loss. We attribute some (but not all) of the performance gain to using training data augmentation, see Sec. 6.6 for a discussion.

RGB + 3D model. When a 3D model is available to render ground truth scene coordinates for training, both DSAC++ and DSAC* benefit, with DSAC* achieving highest accuracy with 85.2% of re-localized frames. Also note that DSAC* is trained more than twice as fast as DSAC++ on identical hardware. *SCoCR* [54] achieves similar accuracy but leverages a more complicated network architecture with multiple, hierarchical classification heads conditioning the scene coordinate regression head as well as higher model capacity (165MB vs. 28MB). Note that *SCoCR* deploys training data augmentation similar to our setup.

RGB-D. When DSAC* estimates poses from RGB-D images, it achieves accuracy comparable to the state-of-the-art approach *OtF Forests* of Cavallari et al. [57]. Cavallari et al. [57] use an ICP and rendering-based post-processing to achieve their top-accuracy. Without post-processing, they report 93.4% on 7Scenes, slightly lower than the accuracy of DSAC*. Note that the difference in accuracy for DSAC* compared to the RGB setups solely stems from the use of Kabsch as a pose solver, since our network still estimates scene coordinates from a grayscale image. The correct depth of image points allows a re-localization pipeline to trivially infer the distance between camera and scene. Note that all RGB-D competitors model the uncertainty of scene coordinates in some form, i.e. predicting full distributions of image-to-scene correspondences. Compared to this, the expressiveness of our framework is limited by only predicting scene coordinate point estimates.

Qualitative Results. We visualize the estimated test trajectory, as well as the pose error, of DSAC* for all scenes and all re-localization settings in Fig. 6. Estimated trajectories are predominately smooth, with outlier predictions concentrated on particular, presumably difficult, areas of each scene. To also visualize the re-localization quality in an augmented reality setup, we compare renderings of 3D models of each scene, using estimated camera poses, with the associated test image in Fig. 7. To give an unbiased impression of the general re-localization quality, we selected the test frame with median pose error for each visualization.

6.3 Results for Indoor Localization (12Scenes)

We report quantitative results for 12Scenes in Fig. 5, right. DSAC* achieves state-of-the-art accuracy in all settings for this dataset, consistently outperforming DSAC++. In general, we would consider this dataset as being solved, with multiple methods achieving an average accuracy of ≈99% for re-localization from RGB-D and RGB images, and in case of DSAC* even from RGB images without a 3D model, the most difficult setting.

6.4 Results for Outdoor Localization (Cambridge)

We measure the re-localization quality on the Cambridge dataset using the median pose error for each scene, see Table 2. Due to the ground truth for this dataset being recovered using a SfM tool, we report results with centimeter precision. We find the expressiveness of millimeter precision dubious given the nature of ground truth poses. Given a 3D model for training, DSAC* and DSAC++ achieve similar accuracy, but DSAC* trains significantly faster. For many scenes, NG-DSAC++ [16], i.e. DSAC++ with neural-guided RANSAC, achieves the best results. In principle, we could extend DSAC* to utilize neural guidance as well. Neural guidance is designed to improve RANSAC in high outlier domains. We expect the benefit of coupling it with DSAC* to be rather small, given the quality of results already.

When training without a 3D model, the new training objective of DSAC* achieves higher accuracy than DSAC++ across all scenes. Notably, DSAC* trained without a 3D model achieves higher accuracy than any method (including DSAC*) trained with a 3D model for the Great Court scene. Great Court is the largest landmark in the dataset. The associated SfM reconstruction contains a high outlier ratio, and might hinder the training process due to its low quality.

We visualize the estimated test trajectories of DSAC* in Fig. 8. Due to the very different scene sizes, we derive a scene-dependent threshold to color-code pose errors. The visualizations reveal that high localization error is correlated with the distance of the camera to the scene, particularly obvious for Old Hospital, but also King's College. In Fig. 9, we depict the median pose error per scene, and observe a high visual quality of re-localization, suitable for augmented reality applications.

6.5 Network Architecture and Runtime

As explained in Sec. 4, we updated the network architecture compared to DSAC++. To disambiguate the impact of the network, and of the updated training schedule, we conduct an ablation study, see Table. 3. We trained both architectures using the updated training schedule of DSAC* on the 7Scenes dataset. Both architectures benefit from the DSAC* training settings. However, the DSAC* architecture combined with DSAC* training achieves the best accuracy, despite faster run time and smaller memory footprint. Together with the stream-lined pose optimization (e.g. using 64 RANSAC hypotheses instead of 256), we achieve a total runtime of the system of 75ms compared to 200ms for DSAC++ on a single Tesla K80 GPU, and 30ms on a GeForce RTX 2080 Ti.

6.6 Impact of Data Augmentation

Li et al. [54] demonstrated the effectiveness of simple geometric training data augmentation (random rotation and re-scaling) for RGB-based camera re-localization (w/ 3D model) on the 7Scenes dataset. We confirm these results here, and show similar effects for RGB-D based re-localization, and RGB-only re-localization. See Fig. 10 for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

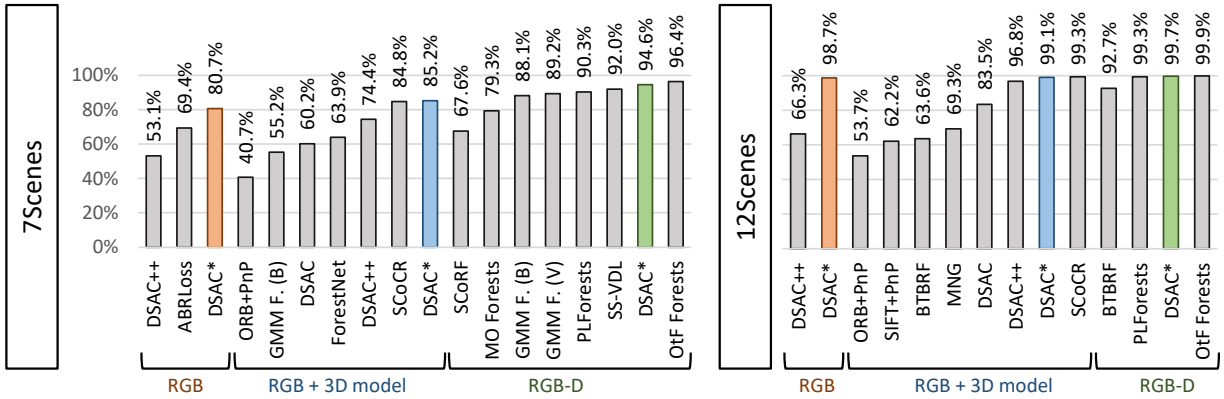


Fig. 5. **Indoor Localization Accuracy.** We report the average percentage of correctly re-localized frames below an error threshold of 5cm and 5° on the 7Scenes [3] and 12Scenes [82] datasets. We group methods by utilized data, i.e. **RGB**: neither a 3D model or depth maps at training and test time, **RGB + 3D model**: a 3D model or depth maps at training time *but not at test time*, **RGB-D**: depth maps at training time *and at test time*. See the main text for references to all methods.

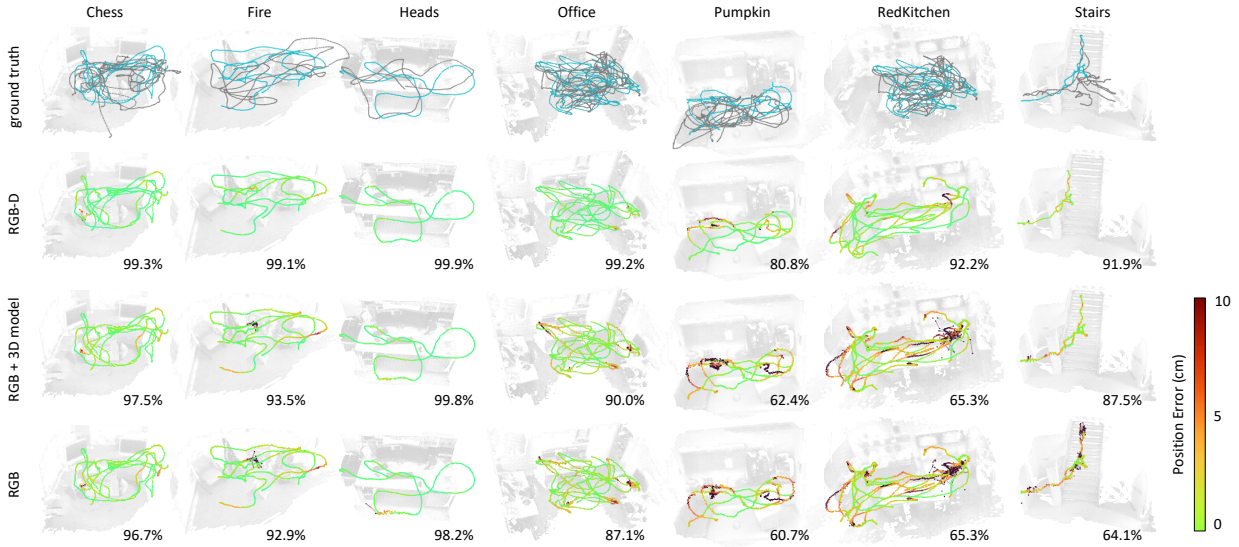


Fig. 6. **Results For Indoor Scenes. First Row:** Camera positions of training frames in gray and of test frames in cyan for all scenes of the 7Scenes [3] dataset. **Remaining Rows:** Estimated camera positions of test frames, color coded by position error. We also state the percentage of test frames with a pose error below 5cm and 5°. Each row represents a different training setup. For a more informative visualization, we show the ground truth 3D scene model as a faint backdrop, and we connect consecutive frames within 50cm tolerance.

TABLE 2

Outdoor Localization Accuracy. We report median errors on the Cambridge Landmarks [26] dataset as translation error (cm) / rotation error (°). *N/A* denotes that a particular result was not reported, whereas a dash (-) indicates that a method does not support this particular setting, i.e. training with or without a 3D model, respectively. Best results in **bold** per column, second best underlined. For DSAC variants we additionally state, in brackets, the training time in days on a Tesla K80 GPU. †On a GeForce RTX 2080 Ti training time of DSAC* reduces to 16h.

		RGB + 3D model					RGB				
Method		Church	Court	Hospital	College	Shop	Church	Court	Hospital	College	Shop
MapNet [29]		-	-	-	-	-	200/4.5	N/A	194/3.9	107/1.9	149/4.2
SpatialLSTM [11]		-	-	-	-	-	152/6.7	N/A	151/4.3	99/1.0	118/7.4
SVS-Pose [28]		-	-	-	-	-	211/8.1	N/A	150/4.0	106/2.8	63/5.7
PoseNet17 [27]		149/3.4	700/3.7	217/2.9	99/1.1	105/4.0	157/3.2	683/3.5	320/3.3	88/1.0	88/3.8
AnchorNet [31]		-	-	-	-	-	104/2.7	N/A	121/2.6	57/0.9	52/2.3
InLoc [25]		18/0.6	120/0.6	48/1.0	46/0.8	11/0.5	-	-	-	-	-
Active Search [7]		19/0.5	N/A	44/1.0	42/0.6	12/0.4	-	-	-	-	-
BTBRF [50]		20/0.4	N/A	30/0.4	39/0.4	15/0.3	-	-	-	-	-
SANet [55]		16/0.6	328/2.0	32/0.5	32/0.5	10/0.5	-	-	-	-	-
DSAC [14]	(4d)	55/1.6	280/1.5	33/0.6	30/0.5	9/0.4	-	-	-	-	-
DSAC++ [15]	(6d)	13/0.4	40/0.2	**20/0.3**	18/0.3	6/0.3	20/0.7	66/0.4	24/0.5	23/0.4	9/0.4
NG-DSAC++ [16]	(6d)	**10/0.3**	**35/0.2**	22/0.4	**13/0.2**	6/0.3	N/A	N/A	N/A	N/A	N/A
DSAC*	(2.5d)†	13/0.4	49/0.3	21/0.4	15/0.3	**5/0.3**	**15/0.6**	**34/0.2**	**21/0.4**	**18/0.3**	**5/0.3**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

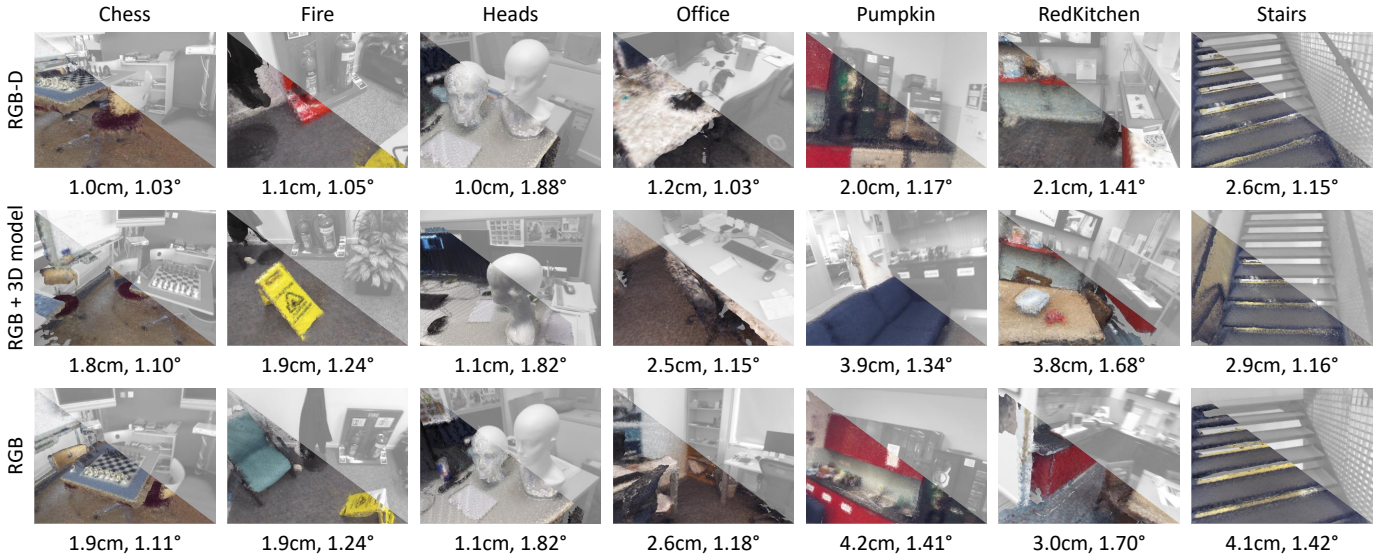


Fig. 7. **Median Errors for Indoor Scenes.** For all test sequences of the 7Scenes [3] dataset, we select the frame with the median pose estimation error. We show the original input frame in gray scale, and a rendered overlay in color using the estimated pose and the ground truth 3D model. We write the associated median pose error below each instance. Each row represents a different training setup.

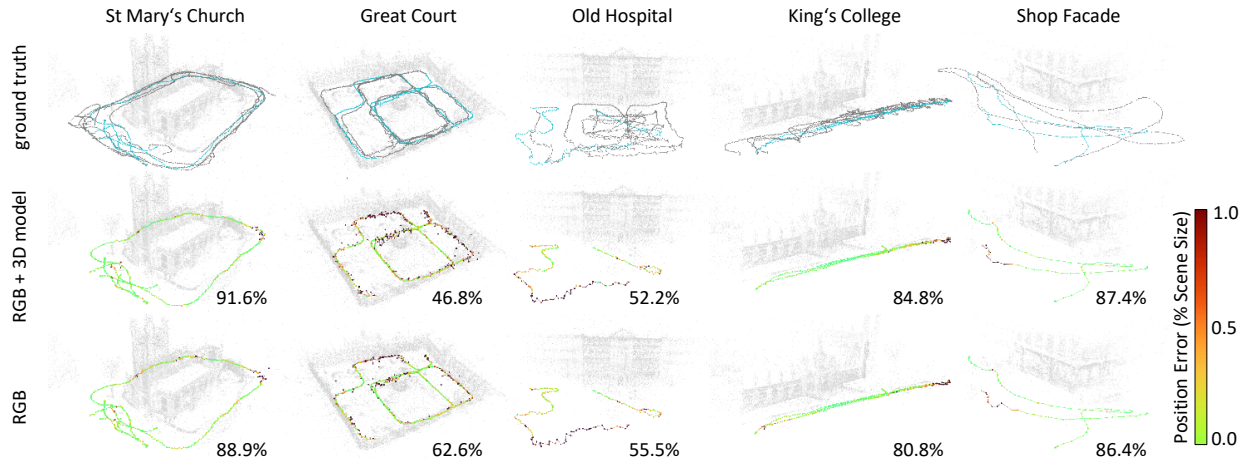


Fig. 8. **Results For Outdoor Scenes. First Row:** Camera positions of training frames in gray and of test frames in cyan for scenes of the Cambridge Landmarks [26] dataset. **Remaining Rows:** Estimated camera positions of test frames, color coded by position error. We also state the percentage of test frames with a position error below 0.5% of the scene size. We derive the threshold for each scene from the scene extent given in [26]. In particular, we use 35cm for St. Mary's Church, 45cm for Great Court, 22cm for Old Hospital, 38cm for King's College and 15cm for Shop Facade. Each row represents a different training setup. For a more informative visualization, we show the ground truth 3D scene model as a faint backdrop, and we connect consecutive frames within 5m tolerance.

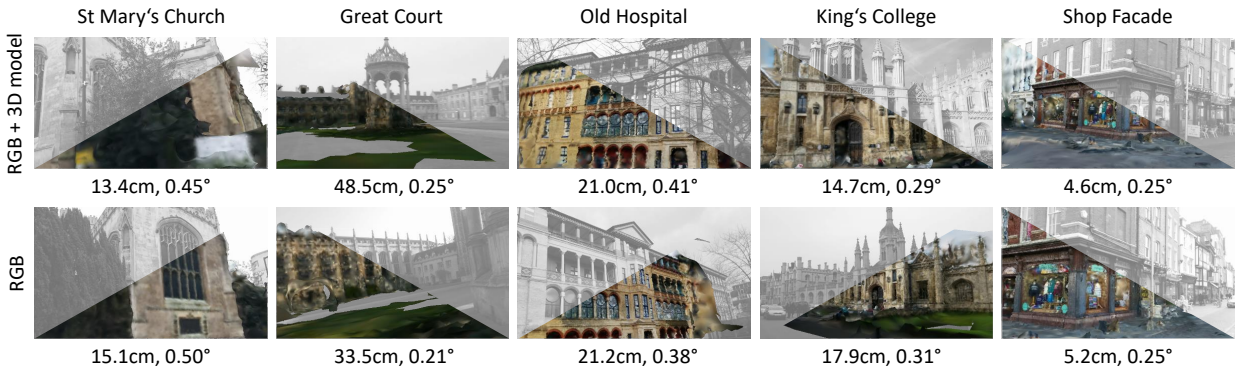


Fig. 9. **Median Errors for Outdoor Scenes.** For all test sequences of the Cambridge Landmarks [26] dataset, we select the frame with the median pose estimation error. We show the original input frame in gray scale, and a rendered overlay in color using the estimated pose and a 3D scene model generated from the ground truth structure-from-motion point cloud. We write the associated median pose error below each instance. Each row represents a different training setup.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3
Comparison of Network Architecture. We compare statistics of the network architecture of DSAC* (this work) and DSAC++ [15]. We train both architectures using the new training schedule of DSAC*. We report accuracy on the 7Scenes dataset for the *RGB + 3D model* setting. Times are for a Tesla K80 GPU or, in brackets, a GeForce RTX 2080 Ti.

Architecture	Size	Time	RF	Training Procedure	Accuracy
DSAC++ (VGGNet)	104MB	150ms	73px	DSAC++	74.4%
				DSAC*	82.0%
DSAC* (ResNet)	28MB	50ms (10ms)	81px	DSAC*	85.2%

quantitative and qualitative results. Data augmentation results in significant improvement for DSAC* on the 7Scenes dataset with +9.1%, +7.7% and +4.1% improvement depending on the setup. We see the largest improvements on the 7Scenes Stairs sequence when training in RGB-only mode (+51.5%), see Fig. 10, right. This scene is dominated by ambiguous, repeating structures and presumably data augmentation helps to resolve some of this ambiguity. On 12Scenes, we observe an improvement for RGB-only re-localization (+8.6%), while the other results on this particular dataset are saturated also without data augmentation. For the Cambridge Landmarks dataset, we found no significant advantage in augmenting the training data. Notably, DSAC* achieves state-of-the-art accuracy across settings also without the use of data augmentation, cf. Fig. 5.

6.7 Impact of the Receptive Field

One important factor when designing an architecture for scene coordinate regression is the size of the receptive field. That is, what image area is taken into account for predicting a single scene coordinate, comparable to the image patch size for sparse feature matching. The architecture of DSAC* has a receptive field size of 81px. By substituting individual 3x3 convolutions with 1x1 convolutions and vice versa (cf. Fig. 3) we can increase and decrease the receptive field and study the change in accuracy. The change of the convolution kernel affects also the total count of learnable parameters of the network. To facilitate conclusions with regard to the receptive field alone, we scale the number of channels throughout the network to keep the number of free parameters constant. We report results in Fig. 11, comparing DSAC* with a receptive field of 81px (standard), 49px and 149px. We observe that the re-localization accuracy decreases with a large receptive field of 149px. While a larger receptive field incorporates more image context for predicting a scene coordinate, is also leads to generalization problems, even when using data augmentation during training. View point changes between training and test set have a higher impact for larger receptive fields. Making the receptive field smaller, with 49px, also decreases accuracy slightly. The effect of having less image context is counteracted by better generalization w.r.t. view point changes. For a more extreme argument in favor of architectures with limited receptive field, we conduct an experiment with an encoder-decoder architecture. Such an architecture encodes the whole image into a global descriptor, and de-convolves it to a full resolution scene coordinate prediction. The receptive field of such an architecture is the whole image, and we ensure again to keep the number of learnable parameters identical. As depicted in Fig. 11 a scene coordinate network with global receptive field achieves a disappointing re-localization accuracy. This indicates, that the receptive field might be another issue connected with the low accuracy of absolute pose regression methods, orthogonal to the explanations given by Sattler et al. in their study of these methods [12].

6.8 Impact of End-to-End Training

We report results before and after training our system in an end-to-end fashion in Fig. 12. For the indoor datasets 7Scenes and 12Scenes we report accuracy using different threshold of 5cm5°, 2cm2° and 1cm1°. While the impact of end-to-end training for a coarse threshold is small, there are significant differences for the finer acceptance thresholds. End-to-end training increases the precision of successful pose estimates, but it does not necessarily decrease the failure rate. We see similar effects in outdoor re-localization for the Cambridge dataset where the pose precision, expressed by the median pose error decreases by ca. 30%. We also provide a qualitative comparison of scene coordinate prediction before and after end-to-end training. Particularly, we visualize areas of training images where the re-projection error increased or decreased due to end-to-end training. The system learns to focus on certain reliable structures. In general, we observe a tendency of the system to increase the scene coordinate quality for close objects. Presumably such objects are more helpful than distant structures for estimating the camera pose precisely.

6.9 Learned 3D Geometry

Scene coordinate regression methods utilize a learnable function to implicitly encode the map of an environment. We can generate an explicit map representation of the geometry encoded in a network. More precisely, we iterate over all training images, predicting scene coordinates to generate one point cloud of the scene. We can recover the color of each 3D point by reading out the associated color at the pixel position of the training image for which the scene coordinate was predicted. Such a point cloud will in general feature many outlier points that hinder visualization. Therefore, we generate a mesh representation using Poisson surface reconstruction [83]. We show the recovered 3D models in Fig. 13 for 7Scenes and in Fig. 14 for Cambridge. Interestingly, our approach learns the complex 3D geometry of a scene, even when training solely from RGB images and ground truth poses. Furthermore, we are able to recover a dense scene representation, even when training with sparse 3D models for the Cambridge dataset.

6.10 Scene Compression Properties

Since scene coordinate regression methods encode the scene geometry within a neural network of fixed capacity, they represent a natural framework for scene compression. In Table 4, we compare the memory demand and accuracy of several learning-based as well as classical re-localization

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

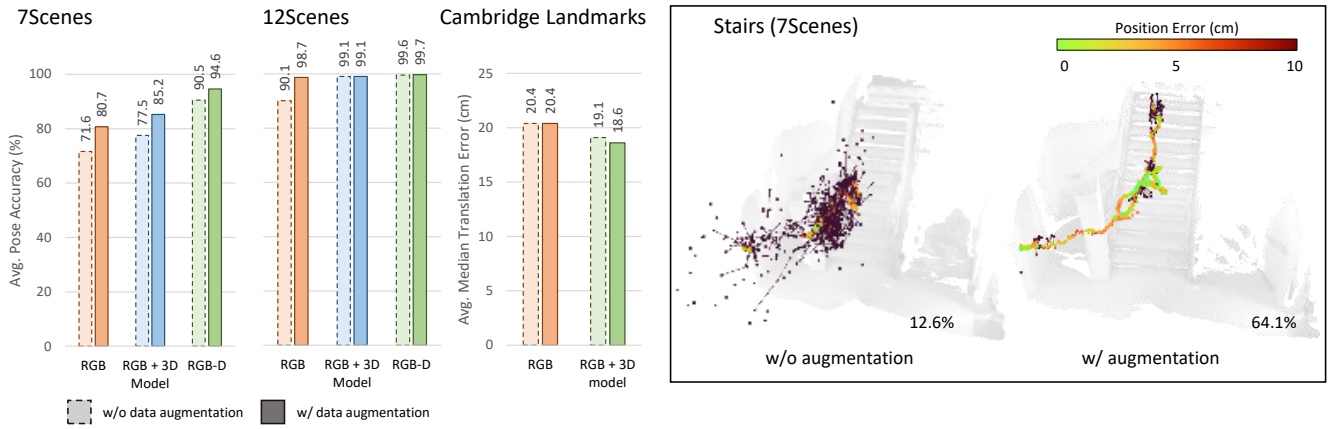


Fig. 10. **Effect of Data Augmentation. Left:** Average percentage of correctly localized frames on 7Scenes [3] and 12Scenes [82], as well as average median translations errors for Cambridge Landmarks [26], with and without using geometric training data augmentation (random rotation and scaling). **Right:** We show qualitative results on the 7Scenes Stairs sequence when training in RGB-only mode (i.e. without using a 3D model of the scene).

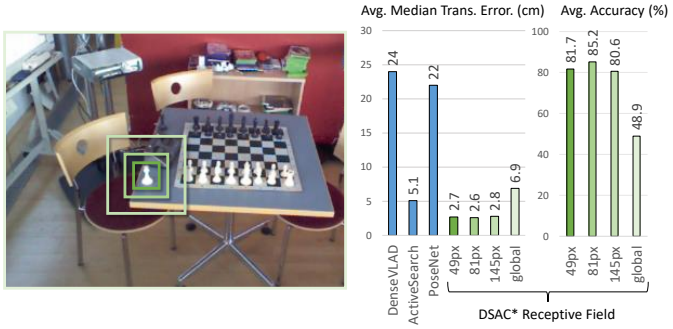


Fig. 11. **Receptive Fields.** We study the impact of the receptive field for DSAC* by altering the underlying network architecture, see the main text for details. **Left.** We visualize the different receptive field sizes of DSAC* relative to a test image. **Right.** We report the median localization errors and percentages of re-localized frames for the 7Scenes dataset. *Global* means a receptive field with the size of the whole image. DenseVLAD [21] and PoseNet [27] also utilize a global receptive field. ActiveSearch [7] utilizes a varying but limited receptive field.

methods. The Cambridge Landmarks [26] dataset is particularly interesting for this comparison, as it features scenes of varying sizes. With a memory footprint of 28MB, DSAC* achieves highest average re-localization accuracy. The retrieval-based DenseVLAD [21] as well as the feature-based hybrid compression schema of Camposeco et al. [84] demand only very little memory but also suffer from low re-localization accuracy. To analyze the scene compression properties of DSAC* further, we re-train our pipeline with a significantly leaner network architecture, called *DSAC* Tiny*. We clamp the number of channels per layer to 128 (cf. Fig. 3) which results in a memory footprint of 4MB per scene. For this analysis, we train using the 3D scene model but without training data augmentation. We found data augmentation to deteriorate results for such a low-capacity network. While *DSAC* Tiny* has a memory demand in the same magnitude as the hybrid compression schema of [84], it achieves significantly higher accuracy. We find that the loss in accuracy compared to the full 28MB model grows with the scene size and complexity, see e.g. the results for *St Mary's Church*. For smaller scenes, such as *Shop Facade*, the loss in accuracy is negligible. We trained *DSAC* Tiny* also for the 7Scenes and 12Scenes datasets, and report an average re-localization accuracy of 73.6% and 98.1%, respectively. Therefore, *DSAC* Tiny* is among the top-performing methods for indoor re-localization despite the small memory demand.

We note that simply increasing network capacity will not enable re-localization on a city scale, as we discuss in [17]. However, DSAC* could be integrated into the large scale re-localization framework of ESAC [17].

7 Conclusion

We have presented DSAC*, a versatile pipeline for single image camera re-localization based on scene coordinate regression and differentiable RANSAC. In this article, we have derived gradients for all steps of robust pose estimation, including PnP solvers. The resulting system supports RGB-D-based as well as RGB-based camera re-localization, and can be trained with or without a 3D model of a scene. Compared to previous iterations of the system, DSAC* trains faster, needs less memory and features low runtime. Simultaneously, DSAC* achieves state-of-the-art accuracy on various dataset, indoor and outdoor, and in various settings. We made the code of DSAC* publicly available, and hope that is serves as a credible baseline in re-localization research.

Acknowledgments

The authors would like to thank Dehui Lin for implementing an efficient version of the differentiable Kabsch pose solver within the scope of his Master thesis.

This work was supported by the DFG grant COVMAP: Intelligente Karten mittels gemeinsamer GPS- und Videodatenanalyse (RO 4804/2-1 and RO 2497/12-2). This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 programme (grant No. 647769).

The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

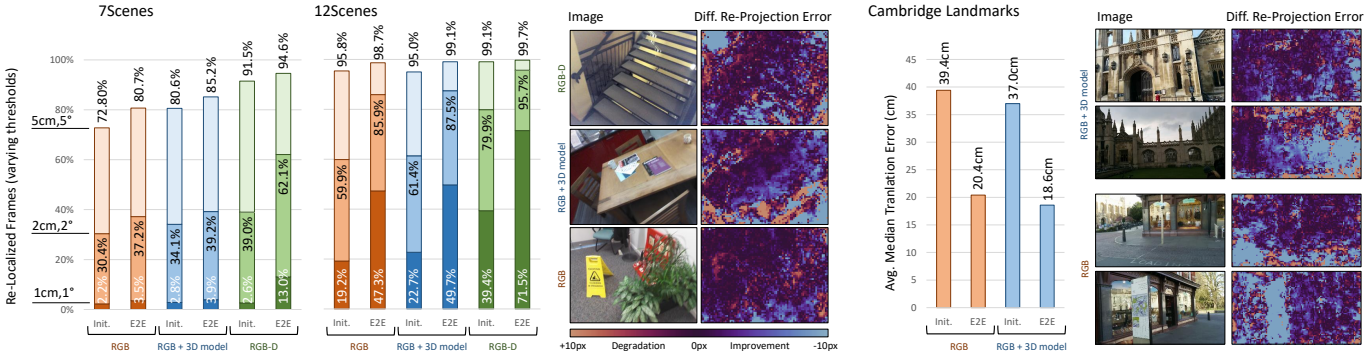


Fig. 12. **Effect of End-to-End Training. Left:** We give the average percentage of correctly localized frames on 7Scenes [3] and 12Scenes [82], before and after end-to-end training, denoted as *Init.* and *E2E*, respectively. We break down accuracy corresponding to pose error thresholds of 5cm/5°, 2cm/2° as well as 1cm/1°. Colors in the bar chart indicate different training setups as also specified at the bottom. Furthermore, we visualize the *difference* in re-projection error before and after end-to-end training for training frames of 7Scenes [3]. Blue indicates that the re-projection error decreased due to end-to-end training, red indicates that the error increased. **Right:** We show the median translation error, averaged over five scenes in Cambridge Landmarks [26], before and after end-to-end training, as well as a visualization of re-projection error change.

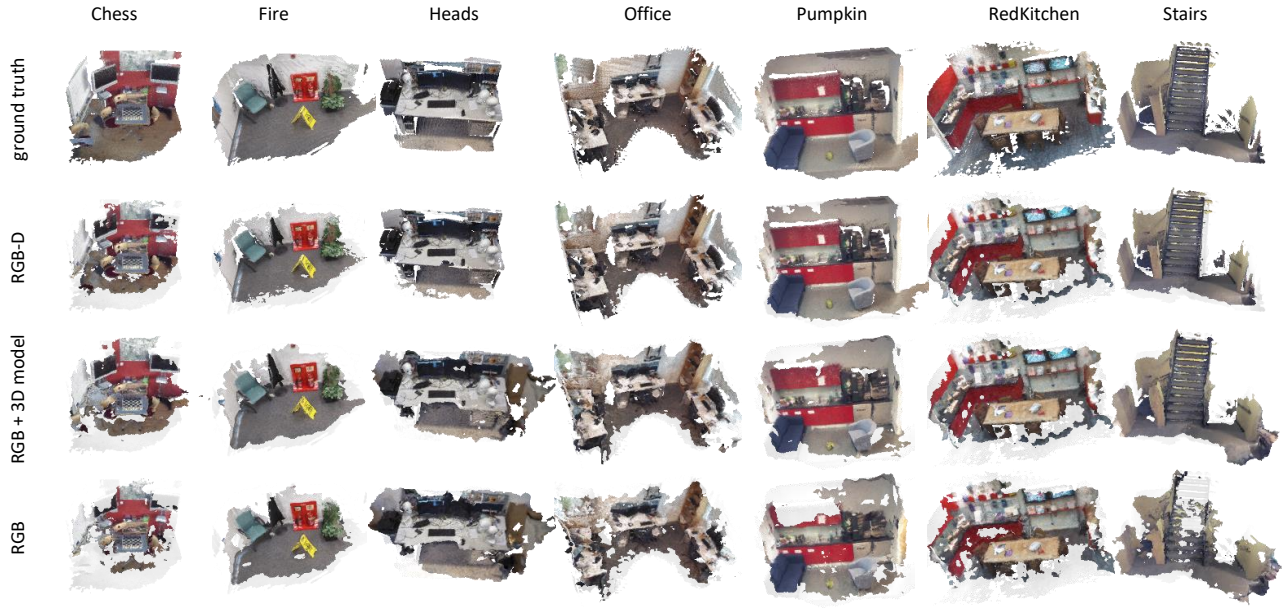


Fig. 13. **Learned Indoor Geometries.** We visualize the 3D scene geometry learned by the scene coordinate regression network for all scenes of the 7Scenes [3] dataset. See the main text for details on how we generated these models. Each row represents a different training setup. In particular, the last row, *RGB*, shows geometry discovered by the network automatically given only RGB images and ground truth poses. For a more informative visualization, we always show the ground truth model as a faint backdrop.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[2] S. Mann, T. Furness, Y. Yuan, J. Iorio, and Z. Wang, "All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality," *arXiv preprint*, 2018.

[3] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *CVPR*, 2013.

[4] J. Valentin, M. Nießner, J. Shotton, A. Fitzgibbon, S. Izadi, and P. H. S. Torr, "Exploiting uncertainty in regression forests for accurate camera relocalization," in *CVPR*, 2015.

[5] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image," in *CVPR*, 2016.

[6] T. Schmidt, R. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *RAL*, 2017.

[7] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *TPAMI*, 2016.

[8] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH*, 2006.

[9] C. Wu, "Towards linear-time incremental structure from motion," in *3DV*, 2013.

[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *CVPR*, 2016.

[11] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization with spatial LSTMs," in *ICCV*, 2017.

[12] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe, "Understanding the limitations of cnn-based absolute camera pose regression," in *CVPR*, 2019.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[14] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC-Differentiable RANSAC for camera localization," in *CVPR*, 2017.

[15] E. Brachmann and C. Rother, "Learning less is more-6D camera localization via 3D surface regression," in *CVPR*, 2018.

[16] ——, "Neural-guided RANSAC: Learning where to sample model hypotheses," in *ICCV*, 2019.

[17] ——, "Expert sample consensus applied to camera re-localization," in *ICCV*, 2019.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2021.3070754, IEEE Transactions on Pattern Analysis and Machine Intelligence

Fig. 14. **Learned Outdoor Geometries.** We visualize the 3D scene geometry learned by the scene coordinate regression network for scenes of the Cambridge Landmarks [26] dataset. See the main text for details on how we generated these models. Each row represents a different training setup. In particular, the last row, *RGB*, shows geometry discovered by the network automatically given only RGB images and ground truth poses. For a more informative visualization, we always show the ground truth model as a faint backdrop. Note that the ground truth models of this dataset are sparse point clouds created by structure-from-motion tools.

	King's College		Old Hospital		Shop Facade		St Mary's Church		**Average**	
	MB used	Median error (m)	MB used	Median error (m)	MB used	Median error (m)	MB used	Median error (m)	MB used	Median error (m)
Active Search [7]	275MB	0.57	140MB	0.52	39MB	0.12	359MB	0.22	203MB	0.36
HybridCompression [84]	**1.0MB**	0.81	**0.6MB**	0.75	**0.2MB**	0.19	**1.3MB**	0.50	**0.8MB**	0.56
DenseVLAD [21]	10MB	2.8	14MB	4.0	3.6MB	1.1	23MB	2.3	13MB	2.6
PoseNet17 [27]	50MB	0.88	50MB	3.2	50MB	0.88	50MB	1.6	50MB	1.6
DSAC++ [15]	104MB	0.18	104MB	**0.20**	104MB	0.06	104MB	**0.13**	104MB	0.14
DSAC*	28MB	**0.15**	28MB	0.21	28MB	**0.05**	28MB	**0.13**	28MB	**0.13**
DSAC* (Tiny)	4MB	0.19	4MB	0.23	4MB	0.07	4MB	0.39	4MB	0.22

TABLE 4

Scene Compression Analysis. We compare memory demand and accuracy of several methods on the Cambridge Landmarks dataset [26]. Information of competitors taken from [84]. DSAC* (Tiny) is a variant of our network architecture with a reduced number of channels per layer.

[20] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *CVPR*, 2007.

[21] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *CVPR*, 2015.

[22] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016.

[23] S. Cao and N. Snavely, "Graph-based discriminative learning for location recognition," in *CVPR*, 2013.

[24] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla, "Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?" in *CVPR*, 2017.

[25] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii, "InLoc: Indoor visual localization with dense matching and view synthesis," in *CVPR*, 2018.

[26] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DoF camera relocalization," in *ICCV*, 2015.

[27] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *CVPR*, 2017.

[28] T. Naseer and W. Burgard, "Deep regression for monocular camera-based 6-DoF global localization in outdoor environments," in *IROS*, 2017.

[29] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *CVPR*, 2018.

[30] V. Balntas, S. Li, and V. A. Prisacariu, "Relocnet: Continuous metric learning relocalisation using neural nets," in *ECCV*, 2018.

[31] S. Saha, G. Varma, and C. V. Jawahar, "Improved visual relocalization by discovering anchor points," in *BMVC*, 2018.

[32] M. Ding, Z. Wang, J. Sun, J. Shi, and P. Luo, "Camnet: Coarse-to-fine retrieval for camera re-localization," in *ICCV*, 2019.

[33] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[34] W. Zhang and J. Kosecka, "Image based localization in urban environments," in *3DPVT*, 2006.

[35] Y. Li, N. Snavely, D. P. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3D point clouds," in *ECCV*, 2012.

[36] L. Svarm, O. Enqvist, M. Oskarsson, and F. Kahl, "Accurate localization and pose estimation for large 3D models," in *CVPR*, 2014.

[37] T. Sattler, M. Havlena, F. Radenovic, K. Schindler, and M. Pollefeys, "Hyperpoints and fine vocabularies for large-scale location recognition," in *ICCV*, 2015.

[38] T. Sattler, M. Havlena, K. Schindler, and M. Pollefeys, "Large-scale location recognition and the geometric burstiness problem," in *CVPR*, 2016.

[39] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson, "City-scale localization for cameras with known vertical direction," *TPAMI*, 2017.

[40] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele, "Real-time image-based 6-dof localization in large-scale environments," in *CVPR*, 2012.

[41] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *ECCV*, 2016.

[42] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *CVPR Workshops*, 2018.

[43] J. Revaud, P. Weinzaepfel, C. R. de Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, "R2D2: repeatable and reliable detector and descriptor," in *NeurIPS*, 2019.

[44] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-Net: A trainable CNN for joint detection and description of local features," in *CVPR*, 2019.

[45] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *CVPR*, 2017.

[46] A. Bhowmik, S. Gumhold, C. Rother, and E. Brachmann, "Reinforced feature points: Optimizing feature detection and description for a high-level task," in *CVPR*, 2020.

[47] A. Guzman-Rivera, P. Kohli, B. Glocker, J. Shotton, T. Sharp, A. Fitzgibbon, and S. Izadi, "Multi-output learning for camera relocalization," in *CVPR*, 2014.

[48] L. Meng, F. Tung, J. J. Little, J. Valentin, and C. W. de Silva, "Exploiting points and lines in regression forests for RGB-D camera relocalization," in *IROS*, 2018.

[49] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1976.

[50] L. Meng, J. Chen, F. Tung, J. J. Little, J. Valentin, and C. W. de Silva, "Backtracking regression forests for accurate camera relocalization," in *IROS*, 2017.

[51] D. Massiceti, A. Krull, E. Brachmann, C. Rother, and P. H. S. Torr, "Random forests versus neural networks - what's best for camera localization?" in *ICRA*, 2017.

[52] X. Li, J. Ylioinas, J. Verbeek, and J. Kannala, "Scene coordinate regression with angle-based reprojection loss for camera relocalization," in *ECCV Workshops*, 2018.

[53] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.

[54] X. Li, S. Wang, Y. Zhao, J. Verbeek, and J. Kannala, "Hierarchical scene coordinate classification and regression for visual localization," in *CVPR*, 2020.

[55] L. Yang, Z. Bai, C. Tang, H. Li, Y. Furukawa, and P. Tan, "SANet: Scene agnostic network for camera localization," in *ICCV*, 2019.

[56] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. Torr, "On-the-fly adaptation of regression forests for online camera relocalisation," in *CVPR*, 2017.

[57] T. Cavallari, S. Golodetz, N. Lord, J. Valentin, V. Prisacariu, L. Di Stefano, and P. H. S. Torr, "Real-time rgb-d camera pose estimation in novel scenes using a relocalisation cascade," *TPAMI*, 2019.

[58] T. Cavallari, L. Bertinetto, J. Mukhoti, P. Torr, and S. Golodetz, "Let's take this online: Adapting scene coordinate regression network predictions for online rgb-d camera relocalisation," in *3DV*, 2019.

[59] O. Chapelle and M. Wu, "Gradient descent optimization of smoothed information retrieval metrics," *Information Retrieval*, 2010.

[60] J. Lee, D. Kim, J. Ponce, and B. Ham, "Sfnet: Learning object-aware semantic correspondence," in *CVPR*, 2019.

[61] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *CVPR*, 2018.

[62] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao, "Learning two-view correspondences and geometry using order-aware network," in *ICCV*, 2019.

[63] R. Ranftl and V. Koltun, "Deep fundamental matrix estimation," in *ECCV*, 2018.

[64] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, "Neighbourhood consensus networks," in *NeurIPS*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018.

[65] T. Probst, D. P. Paudel, A. Chhatkuli, and L. V. Gool, "Unsupervised learning of consensus maximization for 3d vision problems," in *CVPR*, 2019.

[66] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *TPAMI*, 2003.

[67] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[68] K. Levenberg, "A method for the solution of certain problems in least squares." *Quaterly Journal on Applied Mathematics*, 1944.

[69] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, 1963.

[70] G. Bradski, "OpenCV," *Dr. Dobb's Journal of Software Tools*, 2000.

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

[72] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. UIST*, 2011.

[73] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. ISMAR*, 2011.

[74] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *TOG*, 2017.

[75] T. Papadopoulo and M. I. A. Lourakis, "Estimating the jacobian of the singular value decomposition: Theory and applications," in *ECCV*, 2000.

[76] A. Avetisyan, A. Dai, and M. Nießner, "End-to-end cad model retrieval and 9dof alignment in 3d scans," in *ICCV*, 2019.

[77] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *IJCV*, 2009.

[78] W. Förstner and B. P. Wrobel, *Photogrammetric Computer Vision – Statistics, Geometry, Orientation and Reconstruction*, 2016.

[79] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," in *NIPS*, 2015.

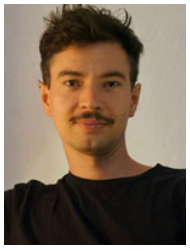
[80] O. Chum and J. Matas, "Randomized ransac with td, d test," in *BMVC*, 2002.

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[82] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," *CoRR*, 2016.

[83] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *SGP*, 2006.

[84] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler, "Hybrid scene compression for visual localization," in *CVPR*, 2019.



Eric Brachmann received the diploma degree with distinction from the TU Dresden (Germany) in 2012. Subsequently, he joined the Computer Graphics Lab and the Computer Vision Lab of the TU Dresden as a research associate under the supervision of Prof. Stefan Gumhold and Prof. Carsten Rother. He received a doctorate in 2018 by the TU Dresden with *summa cum laude*. From 2018 until 2020, he worked as postdoctoral researcher at the Visual Learning Lab of Prof. Rother at the University Heidelberg. In September 2020 he joined Niantic, Inc. as a senior research scientist. He worked on 6D pose estimation of rigid object instances and camera localization in indoor and outdoor scenes. He is an expert in object and scene coordinate regression via machine learning, which is a core element in state-of-the-art learning-based localization techniques. He publishes his work at the leading computer vision conferences (ECCV, ICCV, CVPR), and is an active reviewer (CVPR, ICCV, ECCV, NeurIPS, T-PAMI, IJCV, JMLR). He co-organized two tutorials on visual localization (ECCV, ICCV), and two workshops on 6D pose estimation of objects (ECCV, ICCV). He is interested in constrained machine learning and its combination with traditional computer vision.

Carsten Rother received the diploma degree with distinction in 1999 from the University of Karlsruhe/Germany, conducting his diploma thesis with Prof. Dr. H.-H. Nagel. He received his PhD degree in 2003 from the Royal Institute of Technology Stockholm/Sweden, under the guidance of Jan-Olof Eklundh and Stefan Carlsson. From 2003 until 2013 he was researcher with Microsoft Research Cambridge/UK, and a member of the Computer Vision Group lead by Andrew Blake. From 2014 until 2017 he was full (W3) Professor at TU Dresden. Since September 2017 he is full Professor at Uni Heidelberg, heading the Visual Learning Lab Heidelberg. He is also coordinating director of the Heidelberg Collaboratory for Image Processing (HCI) 3rd phase. His research interests are in the field of computer vision and machine learning – ranging from deep learning and graphical models to smart data generation. He has been working on a broad range of applications – such as image editing (e.g. interactive image segmentation, alpha matting, and deconvolution), image matching (e.g. large displacement Scene Flow), scene understanding (e.g. 6D object pose estimation), Bio-Imaging (e.g. cell tracking). He has published over 150 articles (current H-index 77) at international conferences and journals. He won awards at BMVC '16, ACCV '14, CVPR '13, BMVC '12, ACCV '10, CHI '07, CVPR '05, and Indian Conference on Computer Vision '10. He was awarded the DAGM Olympus prize in 2009. He has co-developed two Microsoft products, GrabCut for Office 2010 and AutoCollage. He also co-authored a book on Markov Random Fields in Computer Vision and Image Processing. He serves as area chair for major conferences and he has been associated editor for T-PAMI.