# Expert Sample Consensus Applied to Camera Re-Localization

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

## Abstract

*Fitting model parameters to a set of noisy data points is a common problem in computer vision. In this work, we fit the 6D camera pose to a set of noisy correspondences between the 2D input image and a known 3D environment. We estimate these correspondences from the image using a neural network. Since the correspondences often contain outliers, we utilize a robust estimator such as Random Sample Consensus (RANSAC) or* Differentiable RANSAC *(DSAC) to fit the pose parameters. When the problem domain,* e.g. *the space of all 2D-3D correspondences, is large or ambiguous, a single network does not cover the domain well.* Mixture of Experts (MoE) *is a popular strategy to divide a problem domain among an ensemble of specialized networks, so called experts, where a gating network decides which expert is responsible for a given input. In this work, we introduce* Expert Sample Consensus *(ESAC), which integrates DSAC in a MoE. Our main technical contribution is an efficient method to train ESAC jointly and end-to-end. We demonstrate experimentally that ESAC handles two real-world problems better than competing methods,* i.e. *scalability and ambiguity. We apply ESAC to fitting simple geometric models to synthetic images, and to camera re-localization for difficult, real datasets.*

## 1. Introduction

In computer vision, we often have a model that explains an observation with a small set of parameters. For example, our model is the 6D pose (translation and rotation) of a camera, and our observations are images of a known 3D environment. The task of camera re-localization is then to robustly and accurately predict the 6D camera pose given the camera image. However, inferring model parameters from an observation is difficult because many effects are not explained by our model. People might move through the environment, and its appearance varies largely due to lighting effects such as day versus night. We usually map our observation to a representation from which we can infer model parameters more easily. For example, in camera re-localization we can train a neural network to predict correspondences between the 2D input image and the 3D environment. Inferring the camera pose from these correspondences is much easier, and various geometric solvers for this problem exist [21, 16, 26]. Because some predictions of the network might be erroneous, *i.e.* we have outlier correspondences, we utilize a robust estimator such as *Random Sample Consensus* (RANSAC) [14], resp. its differentiable counterpart *Differentiable Sample Consensus* (DSAC) [6], or other differentiable estimators [53, 35] for training.

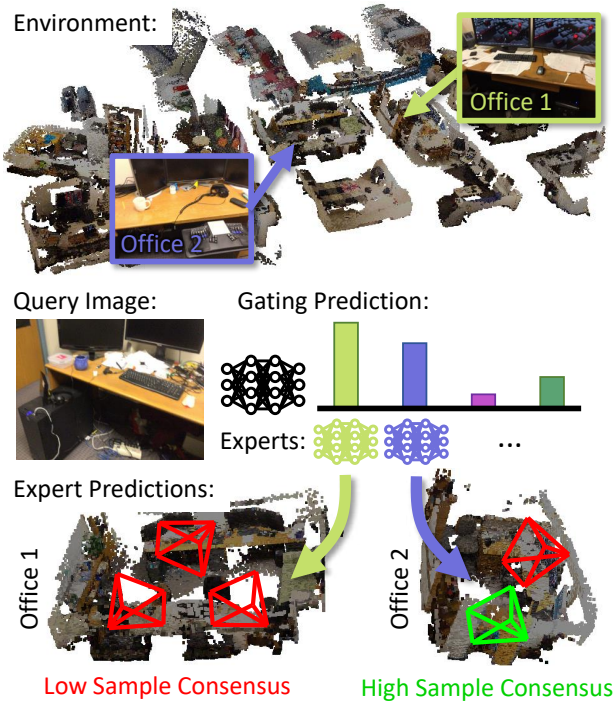


Figure 1. **Camera Re-Localization Using ESAC.** Given an environment consisting of several ambiguous rooms (top) and a query image (middle), we estimate the 6D camera pose (bottom). A gating network (black) predicts a probability for each room. We distribute a budget of pose hypotheses to expert networks specialized to each room. We choose the pose hypothesis with maximum sample consensus (green), *i.e.* the maximum geometric consistency. We train all networks jointly and end-to-end.

For some tasks, the problem domain is large or ambiguous. In camera re-localization, an environment could feature repeating structures that are unique *locally* but not globally, *e.g*. office equipment, radiators or windows. A single feed-forward network cannot predict a correct correspondence for such objects because there are multiple valid solutions. However, if we train an ensemble of networks where each network specializes in a local part of the environment, we can resolve such ambiguities. This strategy is known in machine learning as *Mixture of Experts (MoE)* [20]. Each expert is a network specialized to one part of the problem domain. An additional gating network decides which expert is responsible for a given observation. More specifically, the output of the gating network is a categorical distribution over experts, which either guides the selection of a single expert, or a weighted average of all expert outputs [30].

In this work, we extend Mixture of Experts for fitting parametric models. Each expert specializes to a part of all training observations, and predicts a representation to which we fit model parameters using DSAC. We argue that two realizations of a Mixture of Experts model are not optimal: i) letting the gating network select one expert only [19, 51, 3, 43]; ii) giving as output a weighted average of all experts [20, 1]. In the first case, we ignore that the gating network might attribute substantial probability to more than one expert. We might choose the wrong expert, and get a poor result. In the second case, we calculate an average in model parameter space which can be instable in learning [6]. In our realization of a Mixture of Experts model, we integrate the gating network into the hypothesize-and-verify framework of DSAC. To estimate model parameters, DSAC creates many model hypotheses by sampling small subsets of data points, and fitting model parameters to each subset. DSAC scores hypotheses according to their consistency with all data points, *i.e*. their sample consensus. One hypothesis is selected as the final estimate according to this score. Hypothesis selection is probabilistic, and training aims at minimizing the expected task loss.

Instead of letting the gating network pick one expert, and fit model parameters only to this expert's prediction, we distribute model hypotheses among experts. Each expert receives a share of the total number of hypotheses according to the gating network. For the final selection, we score each hypothesis according to sample consensus, irrespective of what expert it came from, see Fig 1. Therefore, as long as the gating network attributes some probability to the correct expert, we can still get an accurate model parameter estimate. We call this framework *Expert Sample Consensus* (ESAC). We train the network ensemble jointly and end-to-end by minimizing the expected task loss. We define the expectation over both, hypotheses sharing according to the gating network, and hypothesis selection according to sample consensus.

We demonstrate our method on a toy problem where the gating network has to decide which model to fit to synthetic data - a line or a circle. Compared to naive expert selection, our method proves to be extremely robust regarding the gating network's ability to assign the correct expert. Our method also achieves state-of-the-art results in camera re-localization where each expert specializes in a separate, small part of a larger indoor environment.
We give the following main *contributions*:

- We present *Expert Sample Consensus* (ESAC), an ensemble formulation of Differentiable Sample Consensus (DSAC) which we derive from Mixture of Experts (MoE).
- A method to train ESAC jointly and end-to-end.
- We demonstrate the properties of our algorithm on a toy problem of fitting simple parametric models to noisy, synthetic inputs.
- Our formulation improves on two real-world aspects of learning-based camera re-localization, scalability and ambiguity. We achieve state-of-the-art results on difficult, public datasets for indoor re-localization.

## 2. Related Work

**Ensemble Methods.** To improve the accuracy of machine learning algorithms, one can train multiple base-learners and combine their predictions. A common strategy is averaging, so that errors of individual learners cancel out [10, 25, 45, 18]. To ensure that base-learners produce non-identical predictions, they are trained using random subsets of training data (bagging) or using random initializations of parameters (*e.g*. network weights). Boosting refers to a weighted average of predictions where the weights emerge from each base-learners ability to classify training samples [15]. In these ensemble methods, all base-learners are trained on the full problem domain.

In contrast, Mixture of Experts (MoE) [20] employs a divide-and-conquer strategy where each base-learner, resp. expert, specializes in one part of the problem domain. An additional gating network assesses the relevancy of each expert for a given input, and predicts an associated weight. The ensemble prediction is a weighted average of the experts' outputs. MoE has been trained by minimizing the expected training loss [20], maximizing the likelihood under a Gaussian mixture model interpretation [20] or using the expectation-maximization (EM) algorithm [52].

MoE has been applied to image classification where each expert specializes to a subset of classes [51, 19, 1, 3]. Ahmed *et al*. [1] find disjunct subsets by an EM-style algorithm. Hinton *et al*. [19] and Yan *et al*. [51] find subsets of classes based on class confusion of a generalist base network. Aljundi *et al*. [3] apply MoE to lifelong multi-task learning. Whenever their system should be extended with a new task (*e.g*. a new object class) they train a new expert

and a new expert gate. Each expert gate measures the similarity of an input with its associated task, and the gate with the highest similarity forwards the input to its expert.

In all aforementioned methods, the experts' outputs constitute the ensemble output directly. In contrast, we are interested in a scenario where experts output a representation to which we fit parametric models in a robust fashion while maintaining the ability to train the ensemble jointly and end-to-end. To the best of our knowledge, this has not been addressed, previously. Some of the aforementioned methods make use of conditional computation, *i.e*. the gating network selects a subset of experts to evaluate while others stay idle [51, 19, 3]. While this is computationally efficient, routing errors can occur, *i.e*. selection of the incorrect expert results in catastrophic errors. In this work, we distribute computational budget between experts based on the potentially soft prediction of the gating network. Thereby, we strike a good balance between efficiency and robustness.

**Camera Re-Localization.** Camera re-localization has been addressed with a very diverse set of methods. Some authors use image-based retrieval systems [41, 11, 4] to map a query image to the nearest neighbor in a set of database images with known pose. Pose regression methods [23, 50, 22, 5, 9] train neural feed-forward networks to predict the 6D pose directly from an input image. Pose-regression methods vary in network architecture, pose parametrization, or training loss. Both, retrieval-based and pose-regression methods, are very efficient but limited in accuracy. Feature-based re-localization methods [28, 36, 38, 37, 40, 47] match sparse feature points of the input image to a sparse 3D reconstruction of the environment. The 6D camera pose is estimated from these 2D-3D correspondences using RANSAC. These methods are very accurate, scale well but have problems with texture-less surfaces and image conditions like motion blur because the feature detectors fail [44, 23].

Scene coordinate regression methods [44, 17, 49, 7, 31, 32, 6, 12, 33, 8] also estimate 2D-3D correspondences between image and environment but do so densely for each pixel of the input image. This circumvents the need for a feature detector with the aforementioned draw-backs of feature-based methods. Brachmann *et al*. [6] combine a neural network for scene coordinate regression with a differentiable RANSAC for an end-to-end trainable camera re-localization pipeline. Brachmann and Rother [8] improve the pipeline's initialization and differentiable pose optimization to achieve state-of-the-art results for indoor camera re-localization from single RGB images. We build on and extend [6, 8] by combining them with our ESAC framework. Thereby, we are able to address two real-world problems: scalability and ambiguity in camera re-localization. Some scene coordinate regression methods use an ensemble of base learners, namely random forests [44, 49, 7, 31, 32, 12, 33]. Guzman-Rivera *et al*. [17] train the random forest in a boosting-like manner to diversify its predictions. Massiceti *et al*. [31] map an ensemble of decision trees to an ensemble of neural networks. However, in none of these methods do the base-learners specialize in parts of the problem domain.

In [7], Brachmann *et al*. train a joint classification-regression forest for camera re-localization. The forest classifies which part of the environment an input belongs to, and regresses relative scene coordinates for this part. More recently, image-retrieval and relative pose regression have been combined in one system for good accuracy in [46]. Both works, [7] and [46], bear some resemblance to our strategy but utilize one large model without the benefit of efficient, conditional computation. Also, their models cannot be trained in an end-to-end fashion.

**Model Selection.** Sometimes, the model type has to be estimated concurrently with the model parameters. E.g. data points could be explained by a line or higher order polynomials. Methods for model selection implement a trade-off between model expressiveness and fitting error [2, 42]. For illustrative purposes, we introduce ESAC on a toy problem where it learns model selection in a supervised fashion. However, in our main application, camera re-localization, the model type is always known to be a 6D pose.

## 3. Method

We start by reviewing DSAC [6] for fitting parametric models in Sec. 3.1. Then, in Sec. 3.2, we introduce Mixture of Experts [20] with expert selection. Finally, we present ESAC, an ensemble formulation of DSAC in Sec. 3.3. We will explain these concepts for a simple toy problem before applying them to camera re-localization in Sec. 4.

### 3.1. Differentiable Sample Consensus

We are interested in estimating a set of model parameters $\mathbf{h}$ given an observation $I$. For instance, the model could be a 2D line with slope $m$ and intercept $n$, *i.e*. $\mathbf{h} = (m, n)$. Observation $I$ is an image of the line which also contains noise and distractors which are not explained by our model $\mathbf{h}$. See top of Fig. 2 a) for an example input $I$ where the distractors are boxes that partly occlude the line.

Instead of fitting model parameters $\mathbf{h}$ directly to $I$, we deduce an intermediate representation $\mathcal{Y}$ from $I$ to which we can fit our model easily. In the case of a line, $\mathcal{Y}$ could be a set of 2D points $\mathbf{y} \in \mathcal{Y}$ with $\mathbf{y} = (y_0, y_1)$, where each point is explained by our model: $y_1 = my_0 + n$. We can deduce line parameters $\mathbf{h}$ from $\mathcal{Y}$ using linear regression or Deming regression [13].

Since the image formation process is complicated and/or unknown to us, there is no simple way to infer $\mathcal{Y}$ from $I$. Instead, we train a neural network $f$ with learnable parameters $\mathbf{w}$ to predict $\mathcal{Y} = f(I; \mathbf{w})$. The neural network can learn to ignore distractors and image noise to some extent.

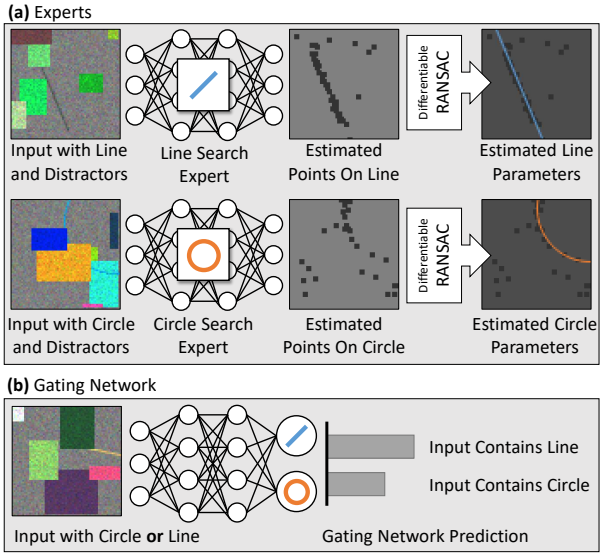


Figure 2. **Network Ensemble for a Toy Problem. a)** Two expert networks, one specialized to finding lines, one specialized to finding circles. Both experts predict a set of 2D points which should lie on the line or circle, respectively. We fit model parameters to these points using differentiable RANSAC. **b)** The gating network predicts whether an image contains either a line or a circle.

However, it is likely to make some mistakes, *e.g.* predict some points $\mathbf{y}$ not explained by our model $\mathbf{h}$. Therefore, we employ a robust estimator $\hat{\mathbf{h}}$, namely Random Sample Consensus (RANSAC) [14], and, for neural network training, Differentiable Sample Consensus (DSAC) [6].

**RANSAC.** RANSAC robustly estimates model parameters by sampling a pool of $N$ model hypotheses $\mathbf{h}_j$ with $j \in \{1, \dots, N\}$. A hypothesis is sampled by randomly choosing a minimal set from $\mathcal{Y}$ and fitting model parameters to it. For a 2D line, a minimal set consists of two 2D points which determine slope and intercept. Each hypothesis is scored by measuring its sample consensus or inlier count $s(\cdot)$, *i.e.* the number of data points $\mathbf{y}$ that agree with the hypothesis.

$$s(\mathbf{h}, \mathcal{Y}) = \sum_{\mathbf{y} \in \mathcal{Y}} \mathbb{1}(\tau - d(\mathbf{y}, \mathbf{h})), \tag{1}$$

where $d(\mathbf{y}, \mathbf{h})$ is a measure of distance between model hypothesis $\mathbf{h}$ and data point $\mathbf{y}$, *e.g.* the point-line distance. Parameter $\tau$ is a threshold that encapsulates our tolerance for inlier errors, and $\mathbb{1}(\cdot)$ denotes the Heaviside step function. Our final estimate is the model hypothesis with the maximum score:

$$\hat{\mathbf{h}} = \mathbf{h}_j \text{ with } j = \underset{j}{\operatorname{argmax}}\, s(\mathbf{h}_j, \mathcal{Y}) \tag{2}$$

Due to the non-differentiability of the argmax selection, we cannot use RANSAC directly in neural network training. However, Brachmann *et al.* [6] proposed a differentiable version of the algorithm which we will discuss next.

**DSAC.** The core idea of Differentiable Sample Consensus [6] is to make hypothesis selection probabilistic. Instead of choosing the hypothesis with maximum score deterministically as in Eq. 2, we choose it randomly according to a softmax distribution over scores:

$$\hat{\mathbf{h}} = \mathbf{h}_j \text{ with } j \sim p(j) = \frac{\exp(s(\mathbf{h}_j, \mathcal{Y}))}{\sum_{j'} \exp(s(\mathbf{h}_{j'}, \mathcal{Y}))} \tag{3}$$

This allows us to minimize the expected task loss $\mathcal{L}(\mathbf{w})$ during training:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{j \sim p(j)} \left[ \ell(\mathbf{h}_j) \right], \tag{4}$$

where $\ell(\mathbf{h})$ measures the error of a model hypothesis $\mathbf{h}$ w.r.t. some ground truth parameters $\mathbf{h}^*$. Since $\mathcal{L}(\mathbf{w})$ is a weighted sum with a finite number of $N$ summands, one for each hypothesis in our pool, we can calculate it and its gradients exactly. As one last consideration, we have to replace the non-differentiable inlier count of Eq. 1 by a soft version [8].

$$s(\mathbf{h}, \mathcal{Y}) = \alpha \sum_{\mathbf{y} \in \mathcal{Y}} 1 - \text{sig}(\beta d(\mathbf{y}, \mathbf{h}) - \beta\tau), \tag{5}$$

where $\text{sig}(\cdot)$ denotes the Sigmoid function, and $\alpha, \beta$ are hyperparameters which control the softness of the score [8].

By minimizing $\mathcal{L}(\mathbf{w})$, we can train our network $f(I; \mathbf{w})$ in an end-to-end fashion using DSAC. The network learns to predict a representation $\mathcal{Y}$ that yields an accurate model estimate $\hat{\mathbf{h}}$, although $\mathcal{Y}$ might still contain outliers. For the toy problem of fitting a 2D line, we show an example run of the full pipeline in Fig. 2 a) top.

### 3.2. Expert Selection

In the following, we introduce the notion of experts for the scenario of parametric model fitting. Firstly, we apply the original formulation of Mixture of Experts (MoE) [20] before extending it in Sec. 3.3.

Instead of training one neural network responsible for all inputs, we train an ensemble of $M$ experts $f_e(I; \mathbf{w})$ with $e \in \{1, \dots, M\}$. We denote the output of each expert with $\mathcal{Y}_e$. A gating network $g(e, I; \mathbf{w})$ decides for a given input $I$ which expert is responsible, *i.e.* it predicts a probability distribution over experts: $p(e) = g(e, I; \mathbf{w})$. For notation simplicity we stack the learnable parameters of all individual networks in a single parameter vector $\mathbf{w}$.

For illustration, we change the toy problem of the previous section in the following way. Some inputs $I$ show a 2D line (as before) while others show a 2D circle. Therefore, we extend our model parameters to $\mathbf{h} = (m, n, r)$. In case of a circle, $(m, n)$ is the circle center and $r$ is its radius. In case of a line, $m$ and $n$ are slope and intercept, respectively and we set $r = -1$ to indicate it is not a circle.

We train two experts, *e.g.* $M = 2$, one specialized for fitting lines, one specialized for fitting circles. Additionally, we train a gating network which should decide for an arbitrary input whether it shows a line or a circle, so that we can apply the correct expert. See Fig. 2 for a visualization of all three networks and their respective task.

Given an image $I$, we first choose an expert according to the gating network prediction $e \sim p(e)$. We let this expert estimate $\mathcal{Y}_e$, and apply DSAC, *i.e.* we sample a pool of hypotheses from $\mathcal{Y}_e$. We choose our estimate similar to Eq. 3 according to

$$\hat{\mathbf{h}} = \mathbf{h}_j \text{ with } j \sim p(j|e) = \frac{\exp(s(\mathbf{h}_j, \mathcal{Y}_e))}{\sum_{j'} \exp(s(\mathbf{h}_{j'}, \mathcal{Y}_e))}. \quad (6)$$

We illustrate the forward process of the ensemble in Fig. 3 a). To train the network ensemble, we can adapt the training formulation of DSAC (Eq. 4) in the following way.

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{e\sim p(e)} \mathbb{E}_{j\sim p(j|e)} \left[\ell(\mathbf{h}_j)\right], \quad (7)$$

*i.e.* we minimize the expected loss over choosing the correct expert according to $p(e)$, and selecting a model hypothesis from this expert according to $p(j|e)$. Note, that we enforce specialization of experts in this training formulation by running the appropriate version of DSAC depending on which expert we chose, *i.e.* we fit either a circle or a line to $\mathcal{Y}_e$.

To calculate the outer expectation, we have to sum over all $M$ experts and run DSAC each time for the inner expectation. Since DSAC is costly, and in some applications we might have a large number of experts, this can be infeasible. However, we can re-write the gradients of the expectation as an expectation itself [6]. This allows us to efficiently approximate the gradients via sampling.

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) &= \mathbb{E}_e \left[ \mathbb{E}_j \left[\ell\right] \frac{\partial}{\partial \mathbf{w}} \log p(e) + \frac{\partial}{\partial \mathbf{w}} \mathbb{E}_j \left[\ell\right] \right] \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \left[ \mathbb{E}_j \left[\ell\right] \frac{\partial}{\partial \mathbf{w}} \log p(e_k) + \frac{\partial}{\partial \mathbf{w}} \mathbb{E}_j \left[\ell\right] \right], \end{aligned} \quad (8)$$

where we sample $e_k \sim p(e)$ $K$ times and average the gradients. We use the abbreviations $\mathbb{E}_e$, $\mathbb{E}_j$ and $\ell$ for the respective entities in Eq. 7. In practice, when training with stochastic gradient descent, we can approximate the expectation with $K = 1$ sample which means that we do one run of DSAC per training input.

Since we select only one expert at test time, we only have to compute this expert's forward pass, which is computationally efficient. However, if we chose the wrong expert, *i.e.* an expert not specialized to current input $I$, we cannot hope to get a sensible prediction $\hat{\mathbf{h}}$. Therefore, the accuracy of this MoE formulation is limited by the accuracy of the gating network. In the next section, we describe our alternative, new formulation which is more robust to inaccuracies of the gating network.

### 3.3. Expert Sample Consensus

Instead of having the gating network select one expert with the risk of selecting the wrong one, we distribute our budget of $N$ model hypotheses among experts. We sample $n_e \leq N$ hypotheses from each expert's prediction $\mathcal{Y}_e$. For this purpose, we define a vector $\mathcal{H}$ that expresses how many hypotheses we assign to each expert.

$$\mathcal{H} = (n_1, \ldots, n_e, \ldots, n_M) \text{ with } \sum n_e = N \quad (9)$$

We choose $\mathcal{H}$ for a given input $I$ based on the output of the gating network. More specifically, $\mathcal{H}$ follows a multinomial distribution based on the gating probabilities $g(e, I; \mathbf{w})$.

$$p(\mathcal{H}) = \frac{N!}{\prod_e n_e!} \prod_e g(e, I; \mathbf{w})^{n_e} \quad (10)$$

Given an image $I$, we first choose $\mathcal{H} \sim p(\mathcal{H})$, and then, according to $\mathcal{H}$ we sample $n_e$ hypotheses $\mathbf{h}_{(e,j)}$ with $j \in \{1, \ldots, n_e\}$ from each expert prediction $\mathcal{Y}_e$. We use an index pair $(e, j)$ to denote which expert a hypothesis belongs to, and which of the $n_e$ hypotheses of this expert it is, specifically. We choose our estimate similar to Eq. 3 and Eq. 6 according to

$$\begin{aligned} \hat{\mathbf{h}} = \mathbf{h}_{(e,j)} \text{ with } (e, j) \sim p(e, j|\mathcal{H}), \text{ and} \\ p(e, j|\mathcal{H}) = \frac{\exp(s(\mathbf{h}_{(e,j)}, \mathcal{Y}_e))}{\sum_{e'} \sum_{j'} \exp(s(\mathbf{h}_{(e',j')}, \mathcal{Y}_{e'}))} \end{aligned} \quad (11)$$

Note that $p(e, j|\mathcal{H})$ is a softmax distribution over all $N$ hypotheses, *i.e.* we choose a hypothesis solely based on its score $s(\cdot)$ irrespective of which expert it came from. In particular, the gating network does not influence hypothesis selection directly, but only guides hypotheses distribution among experts. Depending on the prediction of the gating network $g(e, I; \mathbf{w})$, some experts with low probability will have no hypotheses assigned ($n_e = 0$). For these experts, we do not need $\mathcal{Y}_e$, and hence can save computing the associated forward pass, implementing conditional computation. We visualize our method in Fig. 3 b).

For training, we adapt our MoE training objective of Eq. 7 and minimize

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_{\mathcal{H}\sim p(\mathcal{H})} \mathbb{E}_{(e,j)\sim p(e,j|\mathcal{H})} \left[\ell(\mathbf{h}_{(e,j)})\right]. \quad (12)$$

*i.e.* we minimize the expected loss over distributing $N$ hypotheses, and selecting a final estimate. Since $p(\mathcal{H})$ is a distribution over all possible vectors $\mathcal{H}$, we again rewrite the gradients of $\mathcal{L}(\mathbf{w})$ as an expectation, and approximate via sampling:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}) \approx \frac{1}{K} \sum_{k=1}^{K} \left[ \mathbb{E}_{e,j} \left[\ell\right] \frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}_k) + \frac{\partial}{\partial \mathbf{w}} \mathbb{E}_{e,j} \left[\ell\right] \right] \quad (13)$$

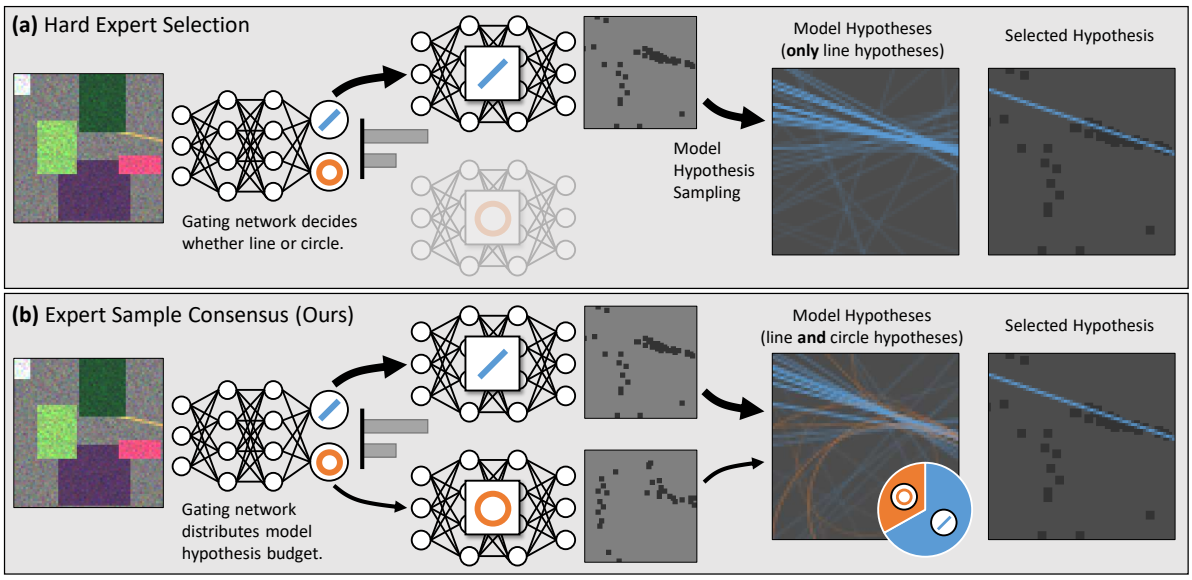


Figure 3. **Ensemble Interplay.** Given an image of a line or a circle, we estimate the parameters of the associated model. **a)** The gating network chooses one expert for a given input. We sample model hypotheses only based on this expert's prediction. **b)** The gating network predicts how the number of model hypotheses should be divided among experts, *i.e.* we sample line **and** circle hypotheses. In this example, the estimate of **a)** and **b)** is similar, but in **b)** we incorporate the full prediction of the gating network, instead of only the largest probability.

In practice we found $K = 1$ to suffice. Throughout training, we sample many different hypotheses splits. Whenever a responsible expert receives too few hypotheses, Eq. 12 yields a large loss, and hence a large training signal for the gating network. On the other hand, receiving too many hypotheses will not decrease the loss further, and there will be no training signal to reward it. Therefore, the gating network learns the trade-off between assigning broad distributions $p(e)$ in ambiguous cases, and assigning sufficiently many hypotheses to the most likely experts.

Calculating the approximate gradients of Eq. 13 involves the derivative of the log probability for a given $\mathcal{H}$ which we calculate as

$$\frac{\partial}{\partial \mathbf{w}} \log p(\mathcal{H}) = \sum_e \frac{n_e}{g(e, I; \mathbf{w})} \frac{\partial}{\partial \mathbf{w}} g(e, I; \mathbf{w}). \qquad (14)$$

## 4. ESAC for Camera Re-Localization

We estimate the 6D camera pose $\mathbf{h} = (\mathbf{t}, \boldsymbol{\theta})$, consisting of 3D translation $\mathbf{t}$ and 3D rotation $\boldsymbol{\theta}$, from a single RGB image. Our pipeline is based on DSAC++ of Brachmann and Rother [8] which itself is based on the scene coordinate regression method of Shotton *et al.* [44]. For each pixel $i$ with 2D position $\mathbf{p}_i$ in an image, we regress a 3D scene coordinate $\mathbf{y}_i$, *i.e.* the coordinate of the pixel in world space.

Given a minimal set of four 2D-3D correspondences $(\mathbf{p}_i, \mathbf{y}_i)$ we can estimate $\mathbf{h}$ using a perspective-n-point algorithm [16, 26]. We employ a robust estimator $\hat{\mathbf{h}}$ as described in Sec. 3. That is, we sample multiple minimal sets to create a pool of $N$ pose hypotheses $\mathbf{h}_j$, and select the best one according to a scoring function. We follow [8], and use a soft inlier count as score. See also Eq. 5 where we use the re-projection error of a scene coordinate for $d(\mathbf{y}, \mathbf{h})$.

Once we have chosen a hypothesis, we refine it using the differentiable pose optimization of [8]. Refinement iteratively resolves the perspective-n-point problem on all inliers of a hypothesis. Gradients are approximated via a linearizion of the objective function in the last refinement iteration. Our output is the refined, selected hypothesis $R(\hat{\mathbf{h}})$. As task loss for training, we use $\ell(\mathbf{h}) = \angle(\boldsymbol{\theta}, \boldsymbol{\theta}^*) + \gamma ||\mathbf{t} - \mathbf{t}^*||$, where $\angle(\cdot)$ denotes angle difference. The hyperparameter $\gamma$ controls the trade-off between rotation and translation errors [23]. We use $\gamma = 100$ when measuring angles in degree and translation in meters.

We estimate scene coordinates $\mathbf{y}$ using an ensemble of experts $f_e(I; \mathbf{w})$ and a gating network $g(e, I; \mathbf{w})$. When designing the expert network architecture we were inspired by DSAC++ [8]. Each expert is an FCN [29] which predicts $80 \times 60$ scene coordinates for a $640 \times 480$px image. Different from DSAC++ [8], we use a ResNet architecture [18] instead of VGG [45]. We found ResNet to achieve similar accuracy while being more efficient in computation time and memory (28 vs. 210MB). Each expert has 16 layers, 6M parameters and a 81px receptive field. The gating network has 10 layers and 100k parameters. The receptive field of the gating network is the complete image, *i.e.* it incorporates more context when assigning experts. Experts have a small receptive field to be robust to view point changes. Our implementation is based on PyTorch [34], and we will make it publicly available[1] .

[1] vislearn.de/research/scene-understanding/pose-estimation/#ICCV19

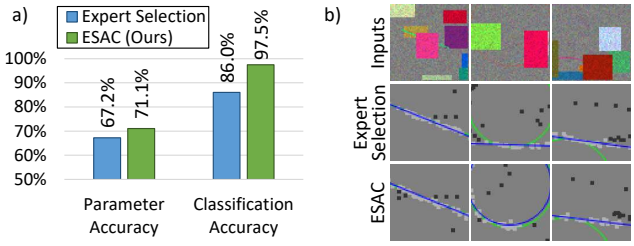


Figure 4. **Results for Toy Problem.** **a)** Percentage of correctly estimated model parameters (left), and percentage of correctly selected model types, *i.e.* line or circle (right). **b)** Qualitative results. The ground truth model is shown in green, the estimate is blue.

## 5. Experiments

We evaluate ESAC for the toy problem introduced in Sec. 3, and camera re-localization from single RGB images.

### 5.1. Toy Problem

**Setup.** We generate images of size 64 × 64px, which show either a line or a circle with 50% probability. We add 4 to 10 distractors to each image, which can occlude the circle or line. Colors of lines, circles and distractors are uniformly random. Finally, we add speckle noise to each image. Difficult example inputs are shown in Fig. 4 b).

We train one expert for lines and one for circles. Each expert is a CNN with 2M parameters that predicts 64 2D points. The gating network is a CNN with 5k parameters that predicts two outputs, corresponding to the probability for a line or a circle. As training loss for lines, we minimize the maximum distance between the estimate and ground truth in the image. For circles, we minimize the distance between centers and absolute difference in radii of the estimate and ground truth. We pre-train each expert using only line or only circle images with DSAC. We pre-train the gating network using both line and circle images with a negative log likelihood classification loss. After pre-training for 50k iterations, we train the ensemble jointly and end-to-end for another 50k iterations, either using *Expert Selection* (Sec. 3.2) or ESAC (Sec. 3.3). We train with a batch size of 32, using Adam [24], and sampling $N = 64$ model hypotheses. For testing, we generate a set of 10,000 images.

**Results.** Fig. 4 a) shows the percentage of correctly estimated model parameters (*Parameter Accuracy*). We accept a line estimate if the maximum distance to the ground truth line in the image is $<$ 3px. We accept a circle estimate if its center and radius is within 3px of ground truth. We observe a significant advantage of using ESAC over Expert Selection (+3.9%). The gating network confuses images with lines and circles sometimes, and might assign higher probability to the wrong expert. ESAC runs both experts in unclear cases, and selects the final estimate according to sample consensus. Fig. 4 a) also shows the classification accuracy of the ensemble, *i.e.* selecting the correct model type. Here, ESAC outperforms Expert Selection by 11.5%. The good classification accuracy indicates that ESAC might be a suitable method for model selection, although we did not investigate this scenario further.

| 7Scenes | Acc. | Med. Err. | 12Scenes | Acc. |
|---|---|---|---|---|
| MapNet [9] | - | 18cm, 6.6° | SIFT+PnP [48] | 62.2% |
| ActiveSearch [38] | - | 5.1cm, 2.5° | BT-RF [32] | 63.6% |
| AC-RF [7] | 55.2% | 4.5cm, 2.0° | MNG [48] | 69.3% |
| DSAC++ [8] | **74.4%** | 3.6cm, **1.1°** | DSAC++ [8] | 96.4% |
| ESAC (Ours) | 73.8% | **3.4cm**, 1.5° | ESAC (Ours) | **97.8%** |

Figure 5. **Pose Accuracy when Scene ID is known.** Percentage of pose estimates with an error below 5cm and 5°, and median errors.

### 5.2. Camera Re-Localization

For our main application, each expert predicts the same model type, a 6D camera pose, but specializes in different parts of a potentially large and repetitive environment.

**Datasets.** The *7Scenes* [44] dataset consists of RGB-D images, camera poses and 3D models of seven indoor rooms (ca. 125m$^3$ total). The images contain texture-less surfaces, motion blur and repeating structures, which makes this dataset challenging despite its limited size. The *12Scenes* [48] dataset resembles 7Scenes in structure but features twelve larger rooms (ca. 520m$^3$ total). The combination of 7Scenes and 12Scenes yields one large environment (*19Scenes*) comprised of 19 rooms (ca. 645m$^3$ total, see also Fig. 1). The data features multiple kitchens, living rooms and offices, containing ambiguous furniture and office equipment.

**Setup.** Ignoring depth channels, we estimate camera poses from RGB only. We train one expert per scene, *i.e.* $M \in \{7, 12, 19\}$ depending on the dataset. We pre-train each expert for 500k iterations, using a $L_1$ regression loss w.r.t. to ground truth scene coordinates obtained by rendering 3D scene models, similar to [8]. Furthermore, we pre-train the gating network to classify scenes using negative log likelihood for 100k iterations. We use Adam with a fixed learning rate of $10^{-4}$. After pre-training, we train the ensemble of networks jointly and end-to-end using Expert Selection (Sec. 3.2) or ESAC (Sec. 3.3) for 100k iterations. We use a learning rate of $10^{-6}$ for experts, and $10^{-7}$ for the gating network. Otherwise, we keep the hyperparameters of DSAC++ [8], *e.g.* we sample $N = 256$ hypotheses and use an inlier threshold of $\tau = 10$px.

**Results on Individual Scenes.** Firstly, we verify our re-implementation of DSAC++, and our choice of network architecture. To this end, we evaluate our expert networks when the scene ID for a test frame is given. That is, we disable the gating network, and always use the correct expert. We achieve an accuracy similar to DSAC++, slightly worse on 7Scenes, slightly better on 12Scenes, see Fig. 5. Note that our networks are 7.5× smaller than those of DSAC++.

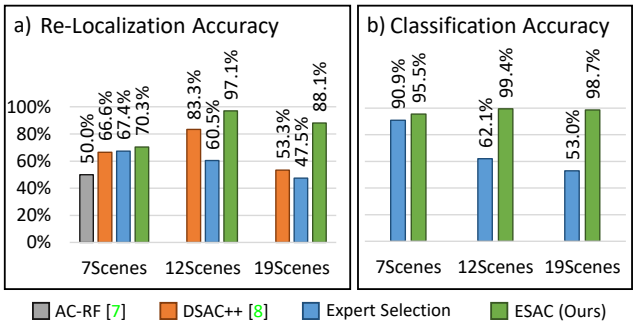


Figure 6. **Average Pose Accuracy when Scene ID is Unknown. a)** Accuracy in growing environments. The scene ID has to be inferred by the method. **b)** Average accuracy of scene classification.

**Results on Combined Scenes.** To evaluate our main contribution, we create three environments of increasing size, combining scenes of 7Scenes, 12Scenes and both (=19Scenes). We compare to DSAC++ by training a single CNN for an environment. For a fair comparison, we use our expert network architecture for DSAC++, and increase its capacity to match that of ESAC's network ensemble. We also compare to an ensemble with Expert Selection (Sec. 3.2). We show our main results in Fig. 6 a) measuring the percentage of estimated poses with an error below 5° and 5cm. The accuracy of DSAC++ decreases notably in larger environments, culminating in a moderate accuracy of 53.3% re-localized images on 19Scenes. DSAC++ relies solely on local image context which becomes increasingly ambiguous with a growing number of visually similar scenes. An ensemble with Expert Selection fares even worse despite using global image context in the gating network when disambiguating scenes. Some of the scenes are too similar, and the top-scoring gating prediction is incorrect in many cases. By distributing model hypotheses among experts, ESAC incorporates global image context in a robust fashion, and consistently achieves best accuracy. The margin is most distinct for 19Scenes, the largest environment, with 88.1% correctly re-localized images. Note that the increased environment scale hardly affects the accuracy of ESAC. It looses 3.5% accuracy for 7Scenes with known scene ID, and less than 1% for 12Scenes, cf. Fig. 5. In the supplement, we include an ablation study about the effect of end-to-end training.

**Handling Ambiguities.** In Fig. 6 b) we show the average scene classification accuracy of Expert Selection and ESAC. In the supplement, we provide additional information in the form of scene confusion matrices, and examples of visually similar scenes. Expert Selection is particularly prone to confuse offices which contain ambiguous furniture and office equipment. ESAC can tell these scenes apart reliably by combining global image context when distributing hypotheses and geometric consistency when selecting hypotheses.

| Method | Dubrovnik [27] Median Accuracy | Aachen Day [39] 0.25m, 2° / 0.5m, 5° / 5m, 10° |
|---|---|---|
| DSAC++ [8] | 2.3°, 24.0m | 0.4% / 2.4% / 34.0% |
| ESAC (10 Experts) | 1.6°, 10.1m | 30.3% / 49.3% / 73.7% |
| ESAC (20 Experts) | **1.4°**, 9.4m | 39.7% / 55.9% / 77.8% |
| ESAC (50 Experts) | 1.6°, 9.1m | 42.6% / 59.6% / 75.5% |
| PoseNet [22] | 4.4°, 7.9m | N/A |
| Active Search [38] | N/A, **1.3m** | **57.3% / 83.7% / 96.6%** |

Figure 7. **Large-Scale Outdoor Re-Localization.** For ESAC, we divide an environment via scene coordinate clustering, and train an expert for each cluster. See the supplement for details.

**Conditional Computation.** By using a single, monolithic network, inference with DSAC++ takes almost 1s on 19Scenes due to the large model capacity. ESAC needs to evaluate only those experts relevant for a given test image. On 19Scenes, it evaluates 6.1 experts in 555ms on average. We can also restrict the max. number of experts per image to trade off accuracy for speed, see the supplement for details.

**Outdoor Re-Localization.** We applied ESAC to outdoor re-localization in vast connected spaces, namely to the Dubrovnik dataset [27], and the Aachen Day dataset [39]. We refer to the supplement for details about the experimental setup, and present the main results in Fig. 7. While we improve over DSAC++ by a large margin, we do not completely close the performance gap to classical sparse feature-based methods like ActiveSearch [38]. We see that adding more experts (and therefore model capacity) helps only to some degree. This hints towards limitations of current scene coordinate regression methods [6, 8] beyond the environment size. For example, the SfM ground truth reconstruction, which we use for training, contains a substantial amount of outliers, particularly for Dubrovnik. The training of CNN-based dense regression might be sensitive to such noisy inputs, and developing resilient training strategies might be a promising direction for future research.

## 6. Conclusion

We have presented ESAC, an ensemble of expert networks for estimating parametric models. ESAC uses a gating network to distribute model hypotheses among experts. This is more robust than formulations where the gating network chooses a single expert only. We applied ESAC to the camera re-localization task in a large indoor environment where each expert specializes to a single room, achieving state-of-the-art accuracy. For large-scale outdoor re-localization, we made progress towards closing the gap to classical, feature-based methods.

**Acknowledgements:** This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No 647769). The computations were performed on an HPC Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden.

## References


[1] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *ECCV*, 2016. 2

[2] Hirotugu Akaike. A new look at the statistical model identification. *TAC*, 1974. 3

[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017. 2, 3

[4] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016. 3

[5] Vassileios Balntas, Shuda Li, and Victor Adrian Prisacariu. RelocNet: Continuous metric learning relocalisation using neural nets. In *ECCV*, 2018. 3

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-Differentiable RANSAC for camera localization. In *CVPR*, 2017. 1, 2, 3, 4, 5, 8

[7] Eric Brachmann, Frank Michel, Alexander Krull, Michael Y. Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016. 3

[8] Eric Brachmann and Carsten Rother. Learning less is more-6D camera localization via 3D surface regression. In *CVPR*, 2018. 3, 4, 6, 7, 8

[9] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *CVPR*, 2018. 3

[10] Leo Breiman. Random forests. *Machine Learning*, 2001. 2

[11] Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In *CVPR*, 2013. 3

[12] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, 2017. 3

[13] William E. Deming. *Statistical Adjustment of Data*. 1943. 3

[14] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 1981. 1, 4

[15] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *IJCAI*, 1999. 2

[16] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003. 1, 6

[17] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014. 3

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6

[19] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Workshops*, 2015. 2, 3

[20] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991. 2, 3, 4

[21] Wolfgang 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. 1

[22] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017. 3

[23] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 3, 6

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[26] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *IJCV*, 2009. 1, 6

[27] Yunpeng Li, Noah Snavely, and Daniel P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010. 8

[28] Hyon Lim, Sudipta N. Sinha, Michael F. Cohen, and Matthew Uyttendaele. Real-time image-based 6-DoF localization in large-scale environments. In *CVPR*, 2012. 3

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 6

[30] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: A literature survey. *Artificial Intelligence Review*, 2014. 2

[31] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, and Philip H. S. Torr. Random forests versus neural networks - What's best for camera localization? In *ICRA*, 2017. 3

[32] Lili Meng, Jianhui Chen, Frederick Tung, James J. Little, Julien Valentin, and Clarence W. de Silva. Backtracking regression forests for accurate camera relocalization. In *IROS*, 2017. 3

[33] Lili Meng, Frederick Tung, James J. Little, Julien Valentin, and Clarence W. de Silva. Exploiting points and lines in regression forests for RGB-D camera relocalization. In *IROS*, 2018. 3

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 6

[35] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *ECCV*, 2018. 1

[36] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *ICCV*, 2015. 3

[37] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *CVPR*, 2016. 3

[38] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 2016. 3, 8

[39] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DoF outdoor visual localization in changing conditions. In *CVPR*, 2018. 8

[40] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3D models really necessary for accurate visual localization? In *CVPR*, 2017. 3

[41] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *CVPR*, 2007. 3

[42] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 1978. 3

[43] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017. 2

[44] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013. 3, 6, 7

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014. 2, 6

[46] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 3

[47] Carl Toft, Erik Stenborg, Lars Hammarstrand, Lucas Brynte, Marc Pollefeys, Torsten Sattler, and Fredrik Kahl. Semantic match consistency for long-term visual localization. In *ECCV*, 2018. 3

[48] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. *CoRR*, 2016. 7

[49] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip H. S. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 3

[50] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization with spatial LSTMs. In *ICCV*, 2017. 3

[51] Zhicheng Yan, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Robinson Piramuthu. HD-CNN: hierarchical deep convolutional neural network for image classification. In *ICCV*, 2015. 2, 3

[52] Bangpeng Yao, Dirk Walther, Diane Beck, and Li Fei-fei. Hierarchical mixture of classification experts uncovers interactions between brain regions. In *NIPS*, 2009. 2

[53] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *CVPR*, 2018. 1